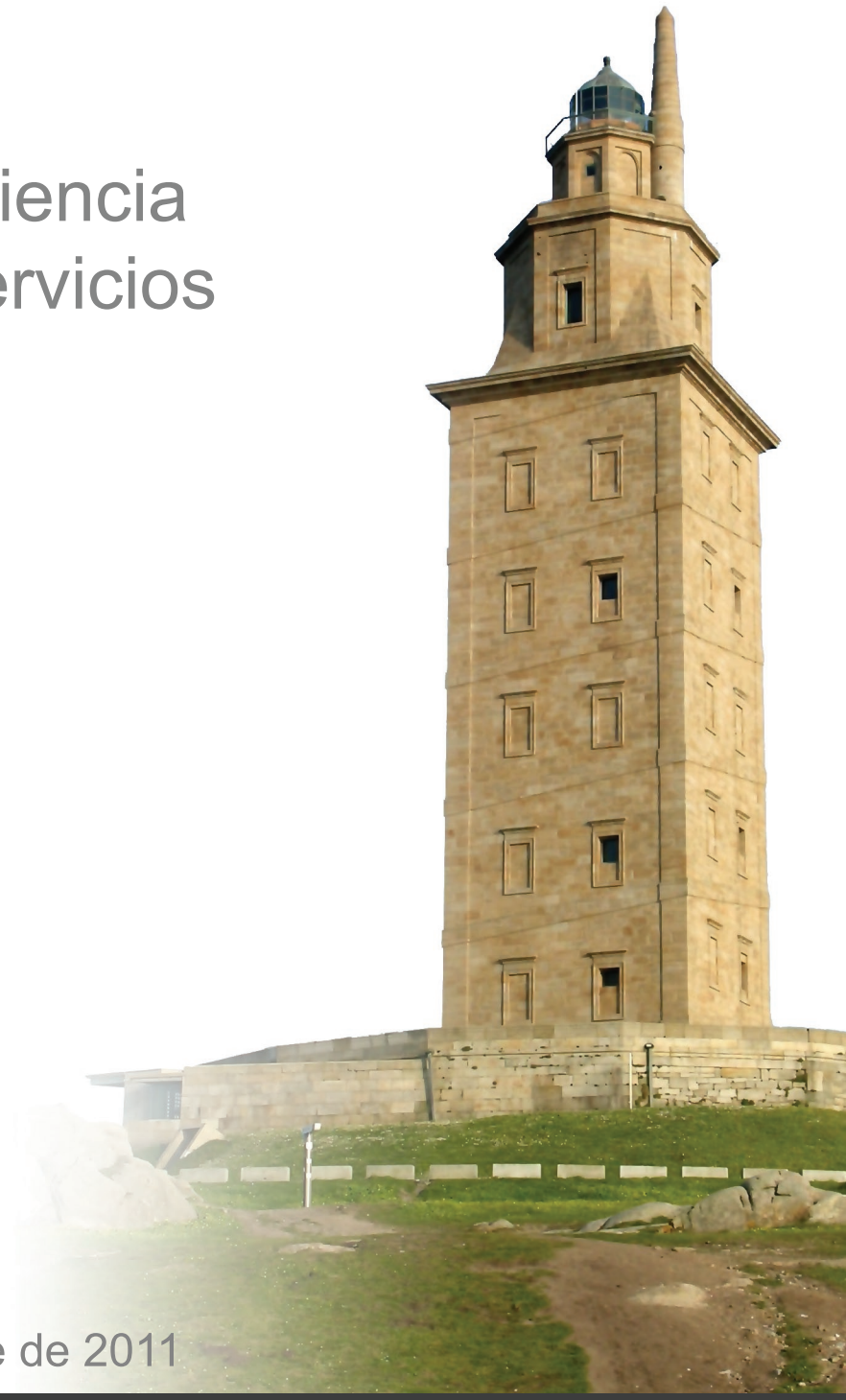
Sistedes
Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software

UNIVERSIDADE DA CORUÑA

Actas de las
VII Jornadas de Ciencia e Ingeniería de Servicios



JCIS 2011

A Coruña, 5-7 de septiembre de 2011

Editores: Pedro Javier Álvarez | José Carlos del Arco | Miguel R. Luaces

VII Jornadas de Ciencia e Ingeniería de Servicios (JCIS)

A Coruña, 5–7 de Septiembre de 2011

Editores:

Pedro Javier Álvarez | José Carlos del Arco | Miguel R. Luaces

Editores:
Pedro J. Álvarez
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universida de Zaragoza
Campus Río Ebro, María de Luna, 1, Edificio Ada Byron, Zaragoza (Spain)
email: alvaper@unizar.es

José Carlos del Arco
Freelance
email: jcarco@gmail.com

Miguel R. Luaces
Departamento de Computación
Facultad de Informática
Universidade da Coruña
Campus de Elviña s/n. 15071, A Coruña (Spain)
email: luaces@udc.es

Servizo de Publicacións da Universidade da Coruña

A Coruña, 2011

CDU: 004: Informática. 004.4: Software

ISBN: 978-84-9749-485-4

Prólogo SISTEDES

La Sociedad de Ingeniería del Software y TEcnologías de DEsarrollo del Software (SISTEDES) ha auspiciado, una vez más, la organización de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD) y las Jornadas sobre Programación y Lenguajes (PROLE). Además, por primera vez este año, la comunidad de Ciencia e Ingeniería de Servicios, vinculada a SISTEDES desde el CEDI 2010, ha celebrado sus jornadas JCIS en unión a JISBD y PROLE. De este modo, este libro de actas se divide en 3 secciones claramente diferenciadas que incluyen los trabajos aceptados en estas tres jornadas.

Así, esta edición de las jornadas JISBD/PROLE/JCIS, celebradas en la Universidade da Coruña, adquiere un especial significado en el panorama de investigación en tecnologías del software, y se constituye en un foro imprescindible, ya que todos los grupos relevantes del estado están vinculados a alguna de estas subcomunidades de SISTEDES. Creemos que la celebración conjunta de los tres eventos facilitará sinergias entre grupos y colaboraciones, y le dará a nuestra investigación una mayor visibilidad social potenciando nuestras posibilidades de transferencia tecnológica a la industria.

Por otro lado, estas jornadas han tenido un carácter de conmemoración de las ***Primeras Jornadas de Investigación y Docencia en Bases de Datos (JIDBD)***, celebradas también en A Coruña precisamente hace 15 años (en 1996). Las JIDBD, unidas a las Jornadas de Ingeniería del software, dieron lugar a las JISBD, que celebraron de modo conjunto su primera edición en Cáceres en 1999.

Desde la organización de estas nuevas y más amplias jornadas SISTEDES, esperamos que este libro de actas os sea de utilidad en vuestro trabajo futuro.

A Coruña, Septiembre 2011
Nieves R. Brisaboa
Víctor M. Gulías
Miguel R. Luaces
Ángeles Saavedra Places

Prólogo

Las Jornadas de Ciencia e Ingeniería de Servicios (JCIS) nacen en el 2011 como resultado de la integración de las anteriores Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB) y del Taller en Procesos de Negocio e Ingeniería de Servicios (PNIS). Esta nueva época supone una ilusión renovada por mantener activa una comunidad formada por expertos en temas estratégicamente claves en el desarrollo socio-económico de cualquier país, como son sin duda los Servicios. Después de siete años, la posibilidad de seguir contando con un foro a nivel nacional de difusión de conocimiento y discusión que nos permita seguir avanzando en estos temas es sin duda una buena noticia para todos. Más alentador aún es comprobar, que año tras año, nuevos grupos de I+D+I de universidades, centros de investigación, empresas o administraciones se incorporan a nuestra comunidad aportando nuevos y valiosos puntos de vista. Este hecho es destacable en el contexto de la crisis económica actual, donde a todos nos supone un esfuerzo extra seguir participando activamente. En esta edición del 2011 hemos querido también estrechar lazos con iniciativas internacionales relacionadas con el tema de los servicios. En este sentido, JCIS se ha convertido en *Service Research & Innovation Institute (SRII) Spain Chapter* como vehículo de promoción de la Ciencia de los Servicios en nuestro país.

A continuación, nos gustaría compartir con vosotros qué ha sido JCIS antes de su apertura y qué esperamos que sea durante su celebración. En la edición 2011 de las Jornadas se han recibido 30 artículos para su revisión, 6 de los cuales habían sido previamente publicados en congresos o revistas internacionales de reconocido prestigio. Después de un arduo proceso de revisión, el Comité de Programa decidió aceptar 23 de estos artículos (75

Desde el punto de vista del programa final, las Jornadas cuentan con dos conferenciantes internacionales de primer nivel: João Falcão e Cunha (Universidade do Porto, Portugal) y Carlos Pedrinaci (London Open University, UK). Ambos conferenciantes tratarán temas de máximo interés para nuestra comunidad como son la aplicabilidad de los servicios a nuestra sociedad actual y la futura Web de datos, respectivamente. Por otro lado, los trabajos aceptados como resultado del proceso de revisión y selección han sido organizados en seis sesiones temáticas dedicadas a la *Composición de servicios*, *Análisis y optimización en BPM*, *Arquitecturas SOA*, *Ingeniería de servicios*, *Procesos de negocio* y, finalmente, *Fundamentos de servicios*. Además de estas sesiones, se celebrarán dos mesas redondas; una de ellas tiene un perfil más académico y está dedicada a *Linked data and services* y la otra aborda los *Avances en gobierno SOA* desde una perspectiva más empresarial. Nos congratula haber conseguido que en estas mesas participen organizaciones científicas y empresas de reconocido prestigio en los temas tratados y poder contar con expertos que tienen amplia experiencia y conocimientos en los mismos.

Antes de finalizar nos gustaría manifestar nuestro agradecimiento más sincero a los autores de los artículos enviados a las Jornadas. Su interés y participación hacen posible la celebración de estas Jornadas. También agradecer al Comité de Programa su disponibilidad, su tiempo y dedicación desinteresada en el proceso de revisión y selección de los artículos recibidos. Su criterio y opinión son claves para garantizar el desarrollo y el éxito de las

Jornadas. También nos gustaría agradecer a los organizadores, el Laboratorio de Bases de Datos de la Universidad de A Coruña, todo el trabajo y esfuerzo dedicado para hacer realidad esta Jornada. Desde el Comité de Programa somos conscientes de las dificultades organizativas de este tipo de eventos, máxime en época de crisis económica, y, por tanto, aplaudimos la labor realizada y las facilidades ofrecidas en todo momento. Estamos seguros del éxito de su celebración. Por último, agradecer también el aval de la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo del Software (SISTEDES), la aportación de nuestros colaboradores (Novatica, ATI, Club-BPM España y Service Research & Innovation Institute) y de aquellos que de alguna forma han puesto su granito de arena para la celebración y éxito de estas jornadas (INES, Isoco y Red Temática española en Linked Data).

Para concluir transmitiros nuestro deseo sincero de que disfrutéis de estas Jornadas y de vuestra estancia en A Coruña. Esperamos volver a veros en 2012.

A Coruña, Septiembre 2011
José Carlos del Arco y Pedro Javier Álvarez (Presidentes del Comité Científico)
Miguel R. Luaces (Presidente del Comité Organizador)

Comité científico

Presidentes:

José Carlos del Arco (Freelance)
Pedro Javier Álvarez (Universidad de Zaragoza)

Miembros:

Amador Durán (Universidad de Sevilla)
Antonio Ruiz-Cortés (Universidad de Sevilla)
Antonio Vallecillo (Universidad de Málaga)
Baltasar Carretero (T-Systems)
Carlos Bobed (Universidad de Zaragoza)
Carlos Rodríguez Fernández (Universidad Complutense Madrid)
Daniel González Morales (Ag. Canaria de Inv., Innovación y Soc. de la Inf.)
Diego López (Red Española de I+D, RedIRIS)
Enrique Álvarez (TCP Sistemas)
Félix García (Universidad de Castilla-La Mancha)
Francisco Almeida Rodríguez (Universidad de La Laguna)
Francisco Javier Fabra (Universidad de Zaragoza)
Francisco Ruiz (Universidad de Castilla-La Mancha)
Guadalupe Ortiz Bellot (Universidad de Extremadura)
Jaime Cid (Oracle)
Jesús Arias Fisteus (Universidad Carlos III de Madrid)
Jesús Contreras (iSOCO)
Jesús Gorroñogoitia (Atos Origin)
Joan Pastor (Universitat Oberta de Catalunya)
Joaquín Peña (Universidad de Sevilla)
Jordi Marco (Universidad Politécnica de Cataluña)
Jorge Cardoso (SAP Research, Germany)
José Emilio Labra (Universidad de Oviedo)
José García Franquelo (Sadiel)
José Hilario Canós (Universidad Politécnica de Valencia)
José M. López Cobo (Playence KG)
José Raúl Romero (Universidad de Córdoba)
Juan De Lara (Universidad Autónoma de Madrid)
Juan Hernández (Universidad de Extremadura)
Juan José Moreno Navarro (Universidad Politécnica de Madrid)
Juan Manuel Murillo (Universidad Extremadura)
Juan Miguel Gómez (Universidad Carlos III de Madrid)
Juan Pavón (Universidad Complutense de Madrid)
Leire Bastida (Tecnalia)
Manuel Lama (Universidad de Santiago de Compostela)

Manuel Resinas (Universidad de Sevilla)
María del Carmen Penadés (Universidad Politécnica de Valencia)
María-Ribera Sancho (Universidad Politécnica de Cataluña)
María Valeria De Castro (Universidad Rey Juan Carlos)
Marta Patiño (Universidad Politécnica de Madrid)
Martín Álvarez Espinar (W3C Spain)
Mercedes Ruiz (Universidad de Cádiz)
Montaña Merchán (Ministerio de Administraciones Públicas)
Óscar Corcho (Universidad Politécnica de Madrid)
Óscar Pedreira (Universidad de A Coruña)
Pere Botella (Universidad Politécnica de Cataluña)
Santi Ristol (Atos Origin)
Silvia Acuña (Universidad Autónoma de Madrid)
Vicente Luque Centeno (Universidad Carlos III de Madrid)
Vicente Pelechano (Universidad Politécnica de Valencia)
Víctor Acinas (Inabensa)
Víctor Ayllón (Novayre)

Comité de organización

Coordinadora:
Nieves Rodríguez Brisaboa
Presidente:
Miguel R. Luaces

Miembros:
José Ramón Paramá Gabía
Antonio Fariña Martínez
Diego Seco Naveiras
Ana Belén Cerdeira Pena
Eduardo Rodríguez López
Sandra Álvarez García
Sergio Folgar Méndez

Índice de contenidos

Sesión 1: Composición de servicios

An optimal and fast algorithm for web service composition

Pablo Rodriguez-Mier, Manuel Mucientes, and Manuel Lama

Centro de Investigación en Tecnologías de la Información (CITIUS)
Universidad de Santiago de Compostela, Spain
{pablo.rodriguez.mier,manuel.mucientes,manuel.lama}@usc.es

Abstract. Automatic composition is not a trivial problem, especially when the number of services is high and there are different control structures to handle the execution flow. In this paper, we present an A* algorithm for automatic service composition that obtains all valid compositions from a extended service dependency graph, using different control structures (sequence, split, choice). A full experimental validation with eight different public repositories has been done, showing a great performance as the algorithm found all valid compositions in a short period of time.

Keywords: Heuristic search; A* algorithm; Web services composition

1 Introduction

Service-oriented computing plays an important role in the development of many areas, such as the improvement of business processes, internet marketing or social networks, and this trend is expected to continue in the coming years as more resources become available. Web services are software components that define formally machine readable interfaces for accessing data through the network. This feature allows to enable greater and easier integration and interoperability among systems and applications.

As a result, in the last year several papers have dealt with composition of web services, which consists in the automatic combination of services in order to combine the functionality of different modules. Some approaches, such as [2, 1], treat the composition problem as a planning problem. In general, these approaches have important drawbacks: high complexity, high computational cost and inability to maximize parallel execution of web services.

Other approaches [4, 6, 3], consider the problem as a search problem, where a search algorithm is applied over a graph or a tree in order to find a minimal composition. These proposals are simpler and more effective than the AI Planners, and also many of them can exploit parallel execution of web services.

This paper is an extension of a previous work [5] that presents a heuristic-based search algorithm to address the problem of the automatic web service composition, which the main contributions are: (1) A optimization techniques to optimize the graph size; (2) A heuristic search based on the A* algorithm

that finds an optimal[1] composition over a service dependency graph and (3) a method to reduce dynamically the possible paths to explore during the search by filtering equivalent compositions. The novelties of our proposal in relation to [5] are: (1) We extended the definition of the service dependency graph to find all solutions with minimal number of services and minimal runpath and (2) we included the use of the *choice* control to support the selection of different alternatives, where more than one service has equivalent services. The rest of the paper is organized as follows: Section 2 explains the service composition algorithm. Section 3 analyzes the algorithm with eight different repositories. Section 4 concludes the paper.

2 Algorithm for web service composition

A web service can be described as a software component that produces a set of outputs given a mandatory set of inputs. Given a request, the composition problem consists in finding a set of services which once executed, produces the desired outputs.

A web service composition, then, can be formulated as a set of web services whose execution is coordinated by a workflow-like structure. This workflow, therefore, has services and a set of control structures that define both the behavior of the execution flow and the inputs/outputs related to those structures. The automatic web service composition algorithm presented in this paper can handle three different control structures: sequence, split and choice.

The present paper is an extension of a previous work [5], in which a heuristic-based algorithm to obtain only the best compositions was presented. Unlike the previous work, the algorithm presented herein is aimed at obtaining all optimal web service compositions in a very short period of time. Our proposal, based on A* algorithm, follows the next steps: 1) Compute a service dependency graph with a subset of services from repository that solves a request and 2) apply the A* search over the generated graph to obtain all valid solutions.

Given a request, an extended service dependency graph (SDG) is dynamically generated. This graph contains a subset of services from the repository that generates the wanted outputs, and it is organized in layers (splits) connected in sequence. Each layer contains one or more services in parallel that can be executed with the outputs generated by the previous layers. The expression for a layer can be defined as follows:

$$L_i = \{S_i : S_i \notin L_j(j < i) \wedge I_{Si} \cap O_{i-1} \neq \emptyset \wedge I_{Si} \subseteq I_R \cup O_0 \cup \ldots \cup O_{i-1}\}$$

where, for each layer L_i: S_i is a service on the i_{th} layer, O_i is the set of outputs generated in the i_{th} layer, I_{Si} is the set of inputs required for the execution of service S_i and I_R is the set of inputs provided by the requester.

The Alg. 1 explains with pseudocode the construction of the graph iteratively. Lines 1-4 initialize the variables used throughout the algorithm. Note that *newOutputs* (outputs generated in the last layer that have not been generated previously) and I_a (available inputs for the current layer) are initialized

[1] A composition with minimal number of services and execution path (runpath).

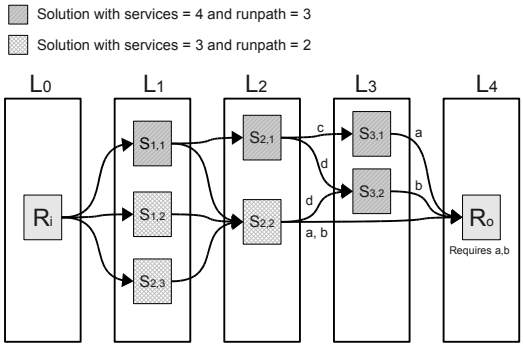


Fig. 1. Example of two solutions with different runpath and different number of services: $sequence(S_{1,1}, S_{2,1}, split(S_{3,1}, S_{3,2}))$ and $sequence(split(S_{1,2}, S_{2,3}), S_{2,2})$

with the same value I_R, as the provided inputs are the first available inputs to the composition and have not been used yet by any service. The main loop starts at line 5. Inside this loop, each layer is calculated following these steps:

1. Obtain all outputs from the previous layers. This outputs are the available inputs to the current layer (L. 7-9).
2. For each service in the repository:
   (a) Check if the service has not appeared in previous layers (L. 12).
   (b) Check if the service can be invoked (i.e. Receives all their inputs from previous layers) (L.13).
   (c) Check if the services uses at least one output that has not been used previously (L. 14).
   (d) If (a), (b) and (c) are true, then the service is added to the current layer.
3. Once all services are selected for the *ith* layer, *newOutputs* is updated by adding the outputs of the *ith* layer and deleting the outputs generated in previous layers. Note that with this operation, only the outputs that have not been used before will remain for the next iteration (L. 19).

With the generation of the graph, the problem of handling *splits* and *sequences* is solved, as the services in the same layer can be invoked in parallel and the connection between layers implies a sequence connection. In order to introduce the *choice* control, we use the advantages of the optimization technique called "Offline service compression" defined in [5]. This technique allows to reduce the dimensions of the SDG by detecting equivalents services from each layer. Using this feature, we can replace all equivalent services by a virtual service. This virtual service holds a reference to each equivalent service, allowing the algorithm to explore less number of solutions. Once the A* algorithm has extracted all solutions, virtual services are replaced by a *choice* control with all equivalent services.

Fig. 1 shows an example of a service dependency graph with five layers and two different solutions. The dark gray services correspond with the services of the

Algorithm 1 Extended service dependency graph algorithm

```
1: newOutputs := I_R
2: I_a := I_R
3: i := 0
4: Layers := ∅
5: repeat
6:    L_i := ∅
7:    for L_j (j < i) do
8:       I_a := I_a ∪ Outputs{L_j}
9:    end for
10:   for Service S_i ∈ Repository do
11:      O_s := Outputs{S_i}
12:      isNewService := S_i ∉ L_j(j < i)
13:      hasInputsAvailable := Inputs{S_i} ⊆ I_a
14:      usesNewInputs := Inputs{S_i} ∩ newOutputs ≠ ∅
15:      if isNewService ∧ hasInputsAvailable ∧ usesNewInputs then
16:         L_i := L_i ∪ S_i
17:      end if
18:   end for
19:   newOutputs := newOutputs ∪ O_s − I_a
20:   Layers := Layers ∪ L_i
21:   i = i + 1
22: until L_i ≠ ∅
```

solution with the largest runpath (the first and the last layers are not computed for the runpath). The two solutions uses parallelism (split) as in both cases, services > runpath. R_i and R_o are dummy services. R_i is a service that provides the request inputs, and R_o is a service that requires as inputs the requested outputs.

Once the SDG is generated, a search over it must be performed in order to find all optimal solutions. For this purpose, we implemented the search using the well-known A* search algorithm. This algorithm will traverse the graph backwards, from the solution (the service whose inputs are the outputs wanted by the requester), to the initial node (the service whose outputs are the provided inputs) handling one service or more in each layer. A detailed explanation of the composition search can be found in [5].

3 Experiments

In order to evaluate the correctness and the performance of the algorithm, a full validation has been done with eight public repositories from Web Service Challenge 2008 (WSC'08)[2]. The repositories have different degree of complexity, having from 158 to 8119 services. The solutions[3] provided by the WSC'08 are

[2] http://cec2008.cs.georgetown.edu/wsc08/downloads/ChallengeResults.rar

[3] The data provided by WSC'08 are not optimal in all cases, as can be seen in the challenge results: http://cec2008.cs.georgetown.edu/wsc08/downloads/WSCResult.pdf

showed in Table 1. The second column indicates the number of services of the repository. The third column shows the minimum number of services required to reach the solution while the fourth column indicates the minimum solution path that can be obtained (according to WSC'08), and the fifth column shows the number of solutions. As can be seen, the complexity of the selected datasets is enough for a complete validation of our proposal.

Table 1. Web Service Challenge: Repository size & provided solutions

Test	Repo. services	Min. #services	Min.exec.path (runpath)	#solutions
WSC'08-1	158	10	3	3
WSC'08-2	558	5	3	4
WSC'08-3	604	40	23	1
WSC'08-4	1041	10	5	2
WSC'08-5	1090	20	8	2
WSC'08-6	2198	40	9	2
WSC'08-7	4113	20	12	2
WSC'08-8	8119	30	20	2

Table 2 shows the results[4] obtained and it is organized as follows: The second column indicates the number of services in the service dependency graph. The third column shows the number of solutions obtained by our algorithm. The fourth column indicates the number of steps executed by the A* search algorithm until the solution was reached. In the fifth column we show the elapsed time until a solution was found (including the time spent in the generation of the service dependency graph). The sixth indicates the number of services obtained by the algorithm and the last one shows the length of the execution path of the solution (runpath). Columns from 4 to 7 are duplicated to show the same information for the solutions with minimal runpath.

As can be seen, in all cases (except in WSC'08-6) the solution with minimal number of services is the solution with minimal runpath too. The first thing that must be noticed is that the solutions obtained by our algorithm are the best for all datasets (according to the solutions provided by WSC'08, see Table 1), except in the case of the dataset WSC'08-6, where our algorithm finds a solution with less number of services (35 vs 40) and a solution with less runpath (7 vs 10), as well as the offered by the WSC'08. Moreover, the algorithm finds all possible solutions for all datasets, showing a great performance as in all cases the bests solutions were found in a very short period of time. This feature is an important advantage over the other approximations since it is the first time that all solutions from the WS-Challenge 2008 (and not only the best solutions) are obtained.

[4] The algorithm was implemented using Java™ JDK 1.6. All the experiments were performed under an Ubuntu 64-bit server workstation (kernel 2.6.32-27) with 2.93GHz Intel® Xeon® X5670 and 16GB RAM DDR-3.

Table 2. Algorithm results for the eight datasets

			Solution with min. Services				Solution with min. Runpath			
Test	Gr.serv	#Sol.	Iter.	Time(ms)	#Serv	Runpath	Iter.	Time(ms)	#Serv	Runpath
WSC'08-1	54	7	20	88	10	3	20	88	10	3
WSC'08-2	55	4	19	161	5	3	19	161	5	3
WSC'08-3	60	1	24	440	40	23	24	440	40	23
WSC'08-4	31	2	11	109	10	5	11	109	10	5
WSC'08-5	81	6	54	498	20	8	54	498	20	8
WSC'08-6	162	12	107	2328	35	14	110	2338	42	7
WSC'08-7	124	2	33	3375	20	12	33	3375	20	12
WSC'08-8	92	3	71	3636	30	20	71	3636	30	20

4 Conclusions and Future Work

In this paper we have presented an extended version of the heuristic-based search algorithm for automatic web service composition. The proposed algorithm allows to find all optimal compositions, with a minimal number of services and minimal runpath. Moreover, a full validation has been done using all datasets provided by the WS-Challenge'08, showing a great performance as our algorithm finds all optimal compositions in a very short time.

As future work we plan to improve our algorithm by including non-functional properties in our model, such as cost, reliability, throughput, etc. Quality of Services (QoS) characteristics are important criteria for building real world compositions. Our algorithm can be easily adapted to handle these features.

Acknowledgment

This work was supported in part by the Dirección Xeral de I+D of the Xunta de Galicia under grant 09SIN065E. Manuel Mucientes is supported by the Ramón y Cajal program of the Spanish Ministry of Science and Innovation.

References

1. K. Chen, J. Xu, and S. Reiff-Marganiec. Markov-htn planning approach to enhance flexibility of automatic web service composition. In *ICWS 2009*, pages 9–16.
2. J. Hoffmann, P. Bertoli, and M. Pistore. Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty. In *AAAI'07*.
3. W. Jiang, C. Zhang, Z. Huang, M. Chen, S. Hu, and Z. Liu. QSynth: A Tool for QoS-aware Automatic Service Composition. In *ICWS 2010*, pages 42–49.
4. S.C. Oh, J.Y. Lee, S.H. Cheong, S.M. Lim, M.W. Kim, S.S. Lee, J.B. Park, S.D. Noh, and M.M. Sohn. WSPR*: Web-Service Planner Augmented with A* Algorithm. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 515–518.
5. P. Rodriguez-Mier, M. Mucientes, and M. Lama. Automatic web service composition with a heuristic-based search algorithm. In *ICWS 2011*. (Accepted).
6. Y. Yan, B. Xu, and Z. Gu. Automatic service composition using and/or graph. In *CEC 2008*, pages 335–338.

Una arquitectura basada en pruebas metamórficas para composiciones de servicios WS-BPEL

Mª del Carmen de Castro Cabrera, Azahara Camacho Magriñán, Inmaculada Medina Bulo y Manuel Palomo Duarte

Escuela Superior de Ingeniería, Universidad de Cádiz,
C/Chile 1, 11002 Cádiz, España
{maricarmen.decastro,inmaculada.medina,manuel.palomo}@uca.es,
azahara.camacmagri@alum.uca.es

Resumen Hoy en día, las composiciones de servicios web y el estándar OASIS WS-BPEL 2.0 juegan un papel importante en los procesos de negocio. Sin embargo, asegurar la validez funcional de este tipo de software representa un reto debido a la escasez de sistemas que automaticen el proceso. Por otro lado, se están utilizando pruebas metamórficas para generar automáticamente casos de prueba que permitan probar programas escritos en lenguajes imperativos tradicionales. Sin embargo, a pesar del éxito conseguido, no han sido aplicadas a composiciones de servicios en WS-BPEL. Este trabajo propone el uso de pruebas metamórficas para la prueba de composiciones WS-BPEL. Se incluye un esquema y descripción de la arquitectura propuesta para su implementación, además de un caso de estudio. Los resultados obtenidos muestran las ventajas que las pruebas metamórficas presentan a la hora de generar automáticamente nuevos casos de pruebas que permitan detectar errores en una composición WS-BPEL.

Keywords: pruebas metamórficas, oráculo, casos de prueba, composiciones de servicios web, WS-BPEL

1. Introducción

Los servicios web (WS) están teniendo, en la actualidad, un gran auge en la sociedad debido al incremento del número de transacciones a través de Internet que se realizan. El lenguaje de ejecución de procesos de negocio de WS, WS-BPEL [20] fue estandarizado por OASIS a petición de las principales empresas del sector TIC (HP, IBM, Oracle, Microsoft, etc.). Este permite desarrollar nuevos WS diseñando procesos de negocio más complejos a partir de otros WS existentes, contando con amplio soporte software para ello [9]. A pesar de ello, su desarrollo no ha ido paralelo a una mejora en las técnicas de prueba para este tipo de software [4]. Una prueba deficiente en un sistema puede dar lugar a errores con consecuencias negativas tanto económicas (especialmente una composición que se invoca de manera masiva [14]) como incluso humanas [15]. Por

tanto, se requieren buenos métodos de prueba que comprueben si las composiciones cumplen los requisitos establecidos. Avances en este sentido se pueden encontrar descritos en [21]. Así mismo, en [11] se presentan dos algoritmos de inferencia para reducir el coste de aplicación de una metodología extendida basada en SOA.

La *prueba metamórfica* (MT) [6] es una técnica de prueba de software que permite generar automáticamente casos de prueba para probar programas. Se basa en el uso de *relaciones metamórficas* (MR), que son propiedades existentes o esperadas definidas sobre un conjunto de entradas y sus correspondientes salidas, para un programa bajo prueba. De hecho, se ha probado con éxito en programas escritos en lenguajes imperativos tradicionales [28]. En cuanto a su efectividad, Zhang [26] ha realizado un experimento en el que compara la capacidad de detección de errores y el coste en tiempo de MT con la técnica de comprobación de asertos estándar. Los resultados muestran que MT detecta más fallos que la citada técnica.

En este trabajo se presenta una propuesta de aplicación de MT para la prueba de composiciones de WS en WS-BPEL. Se incluye un esquema y descripción de la arquitectura propuesta para su implementación, basada en varios sistemas libres ya usados anteriormente por los autores del trabajo. También se incluye un caso de estudio, cuyos resultados permiten mostrar la utilidad de la técnica.

El presente artículo se encuentra estructurado de la siguiente forma. En la sección 2 se explican los conceptos básicos relacionados con MT. A continuación, en la sección 3 se presenta un procedimiento general para implementar MT. La sección 4 muestra el esquema general de la arquitectura propuesta con la descripción de los pasos a seguir para su aplicación. Posteriormente, en la sección 5 se presenta un caso de estudio en el que se muestran las MR diseñadas e implementadas, y los resultados obtenidos. La sección 6 presenta algunos trabajos relacionados con la prueba de composiciones de servicios web con WS-BPEL y con la aplicación de la técnica de prueba MT. Por último, en la sección 7 están las conclusiones y los trabajos futuros.

2. Fundamentos

La prueba de software es una actividad clave en cualquier proyecto de desarrollo de software. La detección de errores y su corrección son actividades fundamentales para asegurar la fiabilidad de un programa. Por este motivo, se han desarrollado diferentes técnicas basadas en diversas propuestas, aunque ninguna ha demostrado ser netamente mejor que las demás [3,19].

2.1. El Problema del Oráculo

Uno de los principales retos a los que se enfrenta la mayoría de las técnicas de prueba de software es el *problema del oráculo*. Un *oráculo* es un mecanismo para comprobar si un programa se comporta correctamente para una entrada

dada. La verificación humana es propensa a errores, y al crecer el número de casos puede ser lenta y costosa.

Mientras que la verificación automática no es trivial, pues se necesitaría una versión implementada del programa (esto sería el programa en sí), quizás con los mismos requisitos funcionales que el original, aunque eliminando restricciones de eficiencia y otros requisitos no funcionales. Sin embargo, esto no es viable en sistemas complejos y reduce el problema a la corrección de otro programa más sencillo (cuya corrección no suele ofrecer las mismas garantías).

Es más, ni siquiera es posible siempre. A veces, el resultado de un programa puede ser difícil de verificar porque no se conoce a priori, porque su tamaño haga imposible la aplicación de cualquier técnica tradicional o simplemente porque debido a su naturaleza, algunos programas son inherentemente difíciles de probar. Por ejemplo, esto suele ocurrir con programas que implican cálculos complejos, tales como aplicación de ecuaciones en derivadas parciales, o la gestión de ingentes cantidades de datos, tales como la gestión de bases de datos y servidores web. En estos casos, es recomendable utilizar propuestas alternativas para el proceso de verificación y validación.

Estas razones avalan el uso de MT para aliviar el problema del oráculo [6,7].

2.2. Pruebas Metamórficas (MT)

MT está basada en la noción de MR. En [2], las MR se definen como "propiedades esperadas o existentes sobre una serie de entradas diferentes y sus correspondientes resultados para evaluaciones múltiples de una función objetivo".

Cuando la implementación es correcta, es de esperar que las entradas y salidas del programa cumplan algunas propiedades necesarias que son relevantes para los algoritmos subyacentes. Estas propiedades son conocidas tradicionalmente como *asertos*. En este sentido, una MR es un tipo de aserto que se expresa en función de los valores de un caso de prueba (es decir, una entrada de un programa y la salida esperada).

Pero, además, una MR debe proveer una forma de generar nuevos casos de prueba a partir de los suministrados inicialmente. Para ilustrar esto, consideremos un programa de ordenación, *ordena*, que ordena una lista de enteros dada como entrada, resultando una lista con los mismos valores pero ordenados ascendentemente.

Por ejemplo, cuando tenemos como entrada la lista $l_1 = \langle 4, 7, 2\rangle$, el resultado esperado de $ordena(l_1)$ es $\langle 2, 4, 7\rangle$. Por tanto, si utilizamos una función, *perm*, que permuta elementos de una lista dada para generar un caso de prueba nuevo en la entrada, podemos asegurar que su resultado tiene que ser el mismo (en este caso, la condición sobre la salida es la función igualdad).

Siguiendo con el mismo ejemplo, sea $l_2 = perm(l_1, \langle(1, 2)\rangle) = \langle 7, 4, 2\rangle$ (sólo hay un intercambio entre el primer y el segundo elemento), el resultado esperado de $ordena(l_2)$ tiene que ser el mismo que el de $ordena(l_1)$. Esto es, $ordena(l_1) = ordena(l_2)$. Por tanto, podemos formalizar esta propiedad, MR_1, de la siguiente forma:

$$MR_1 \equiv (\exists x\ l_2 = perm(l_1, x)) \rightarrow ordena(l_2) = ordena(l_1)$$

donde l_1 es la entrada original y l_2 será el *caso de prueba siguiente* (la permutación x-ésima de l_1). Es preciso tener en cuenta que *perm* es la función de generación de casos de prueba asociada a MR_1, que recibe una lista de entrada (el caso de prueba inicial) y una permutación (una lista de pares de índices), y produce una nueva lista (el caso de prueba siguiente). Si la propiedad metamórfica no se cumple entonces el programa contiene un error.

3. Implementación de MT

La generación de casos de prueba puede ser automatizada con técnicas tradicionales. Volviendo al ejemplo de la sección anterior, si reemplazamos el segundo parámetro de *perm* por una permutación aleatoria obtendríamos la prueba aleatoria tradicional para este ejemplo. Naturalmente, una sola MR es, en general, insuficiente. En el ejemplo de la sección anterior, podríamos no detectar ciertos fallos con MR_1, ya que la corrección para la ordenación implica el uso de la permutación (la lista resultante tiene que ser una permutación de la original) y una restricción de orden (tiene que estar ordenada).

Así, MT es una técnica de prueba que comienza con un conjunto de casos de prueba previo (que será producido con alguna estrategia de generación) y un conjunto de MR. Tras la ejecución del programa, obtenemos que algunos casos de prueba detectan errores y otros no. Los primeros, hacen que el programa sea revisado y el error corregido y, los segundos, llamados *casos de prueba exitosos*, serán seleccionados como entrada a la arquitectura que vamos a utilizar. Entonces, el conjunto de MR se utiliza para generar los *casos de prueba siguientes* a partir de este conjunto exitoso. Estos casos de prueba siguientes formarán el conjunto de casos de prueba (inicial) de la siguiente iteración y el proceso se repetirá.

Por tanto, una vez generado un conjunto de casos de prueba previo (con una estrategia de generación) para aplicar con éxito MT hay que llevar a cabo los siguientes pasos, según se observa en la figura 1:

1. Escoger los casos de prueba que no resulten erróneos (casos de prueba exitosos). Estos conforman el *conjunto de casos de prueba inicial*.
2. Seleccionar adecuadamente las MR teniendo en cuenta el problema a resolver y la estructura del algoritmo empleado en el programa.
3. Generar el conjunto de casos de prueba siguientes aplicando las MR al conjunto de casos de prueba inicial.
4. Ejecutar el programa con los casos de prueba inicial y siguiente.
5. Comparar los resultados.
6. Mejorar el programa corrigiendo los erorres detectados, seleccionar nuevos casos de prueba y/o nuevas MR mejoradas para la siguiente iteración.

4. Arquitectura Propuesta para aplicar MT en WS-BPEL

Una vez que hemos analizado los diferentes aspectos relacionados con MT, en esta sección se propone una arquitectura para su aplicación a composiciones de servicios web WS-BPEL.

La arquitectura integra sistemas de código abierto utilizados desde hace varios años por lo autores: ActiveBPEL como motor de ejecución WS-BPEL y BPELUnit como biblioteca para las pruebas unitarias. ActiveBPEL es el motor para la ejecución de composiciones WS-BPEL 2.0 liberado como código abierto. Es una aplicación ligera, lo que reduce el tiempo necesario para la ejecución del conjunto de casos de prueba. Su mantenimiento es llevado a cabo por Active-VOS [1], que ofrece productos comerciales basados en esta herramienta. BPELUnit es una biblioteca de pruebas unitarias para WS-BPEL [16], que puede ser usada para cualquier motor de WS-BPEL 2.0. Opcionalmente pueden reemplazar a servicios externos con *mockups* (servicios simulados) que implementan el comportamiento indicado en el caso de prueba. Nótese que sólo se recomienda usar esta funcionalidad si no están disponibles los WS que la composición invoca, si se quiere controlar el comportamiento de servicios cuyas respuestas cambien en el tiempo (como valores de bolsa, predicciones meteorológicas, etc.) o si simplemente desea probarse su comportamiento ante determinadas situaciones (un servicio da una respuesta concreta, otro no está disponible, etc.).

Nuestra propuesta trata de mejorar el conjunto de casos de prueba, a partir de las MR, generando nuevos casos de prueba, que permitan detectar mayor número de errores en las composiciones.

A continuación describimos cada una de las etapas de la arquitectura propuesta, cuyo esquema podemos observar en la figura 2:

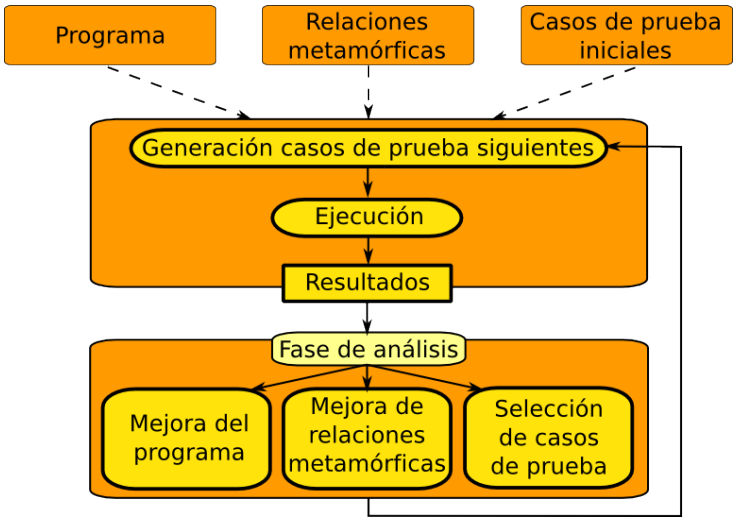


Figura 1: Diagrama de implementación de MT

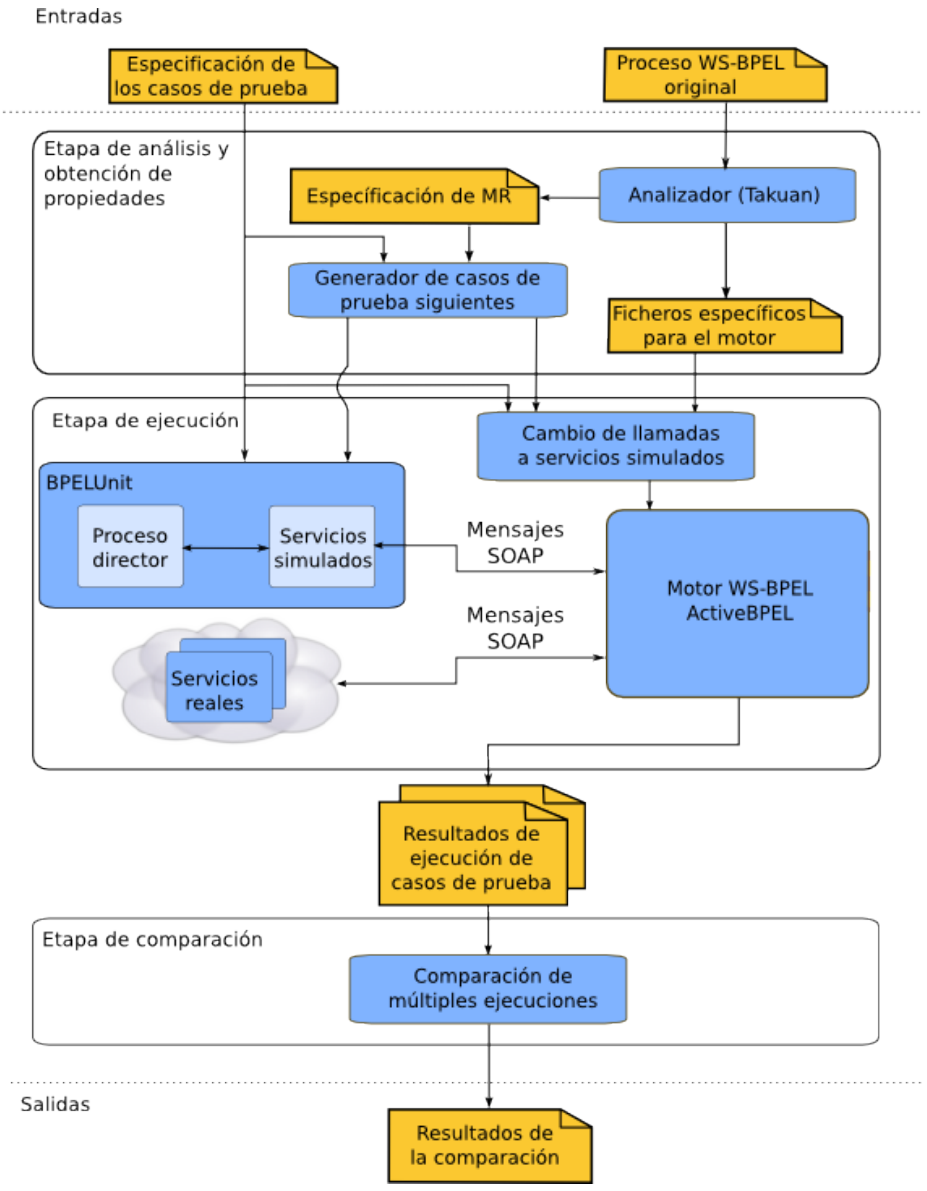


Figura 2: Arquitectura para la aplicación de MT en WS-BPEL

1. *Etapa de análisis y obtención de propiedades.* Se trata de analizar la composición para obtener propiedades importantes y especificar las MR. En este paso podemos apoyarnos en nuestra herramienta libre Takuan [22], que detecta propiedades en una composición BPEL a partir de un conjunto de casos

de prueba. También se generan casos de prueba siguientes aplicando las MR especificadas. Este paso recibe como entradas: la composición WS-BPEL y un conjunto de casos de prueba generados de forma aleatoria o mediante una herramienta automática. Como resultado de esta etapa y necesario para el siguiente paso, se obtiene un conjunto de casos de prueba siguientes (generados a partir de MR) y los ficheros específicos de ActiveBPEL precisos para poder ejecutar la composición. Los módulos principales son:

- Un analizador, que permite especificar las MR adecuadas a la composición, y que prepara los ficheros específicos para la ejecución de la composición.
- Un generador de casos de prueba siguientes, que implementa las MR y obtiene nuevos casos de prueba a partir de los iniciales y las MR implementadas.

2. *Etapa de ejecución.* En esta etapa se ejecuta la composición bajo los casos de prueba iniciales y siguientes. Cada caso de prueba contiene un mensaje inicial que provocará que el servidor lance una instancia del proceso WS-BPEL y las respuestas que suministrarán los servicios externos que queremos sutituir. Como entrada a este paso se reciben los ficheros específicos para el motor ActiveBPEL y los conjuntos de casos iniciales y siguientes obtenidos en el paso anterior. Como salida obtenemos los resultados de la ejecución de los casos de prueba. Los módulos principales son:
   - Un motor WS-BPEL que ejecuta la composición llamando a los servicios correspondientes (reales o simulados) según sea necesario. En concreto, el motor propuesto es ActiveBPEL [1].
   - Una biblioteca de pruebas unitarias de WS-BPEL que se encarga de desplegar el proceso, enviar una petición inicial que da lugar a una nueva instancia del proceso. Esta debe disponer de un proceso director que despliegue el proceso y los servicios simulados que sean necesarios. El entorno BPELUnit [16], que no depende de un motor WS-BPEL específico, cumple estos requisitos.
3. *Etapa de comparación.* Esta etapa recibe los resultados de ejecutar los casos de prueba (iniciales y siguientes). Trata de comparar los resultados de la ejecución de los casos iniciales y los siguientes, con el objetivo de detectar errores en la composición. Básicamente, se comprueba si se cumplen las MR que han generado los casos de prueba siguientes, ya que estas relaciones son propiedades necesarias. En caso de que no se cumplan, se ha detectado un error en la composición. Los resultados de esta comparación permiten decidir el número de iteraciones del proceso según una condición establecida.

5. Caso de estudio

En esta sección aplicaremos la arquitectura propuesta a una composición de ejemplo. En concreto, usaremos la composición del *Préstamo* (Loan Approval), definida en [20]. Básicamente, la composición consiste en la petición por parte de un cliente de una cantidad de dinero a un servicio de préstamo bancario que,

en función de si es menor que 10000 euros y del riesgo asignado al cliente por un servicio asesor, decide aprobarlo o consultar un servicio aprobador, dando como respuesta al cliente si el préstamo se acepta o no. La lógica de la composición la podemos ver en la figura 3.

La primera fase consiste en analizar la composición y los casos de prueba originales para diseñar MR adecuadas a ellos. Por tanto, lo primero que hay que realizar es un estudio de los casos de prueba originales y el diseño e implementación de MR adecuadas a esta composición y a estos casos de prueba.

Para ello, vamos a comentar primero una serie de ejemplos en los que a partir de unos casos de prueba originales, obtenemos unos nuevos casos aplicando las MR correspondientes. Es preciso tener en cuenta que los casos de prueba se extienden con las respuestas de los servicios, es decir, no se trata sólo de entradas y salidas, sino que hay que considerar las respuestas de los servicios implicados. En primer lugar debemos saber los elementos de los que consta un caso de prueba, que son:

- *req_amount*, representa la cantidad de dinero que se pide en el préstamo.
- *ap_reply*, representa la respuesta del aprobador.
- *as_reply*, representa la respuesta del asesor.
- *accepted*, que representa la decisión que se ha tomado finalmente en este caso de prueba, es decir, si el préstamo se concede o no.

Pues bien, vamos a centrarnos en los valores que toman *req_amount*, *ap_reply*, *as_reply* y *accepted*. Para ver más claramente la estructura, aquí mostramos cómo se organizarían estos valores dentro de un caso de prueba teórico:

(*req_amount*, *ap_reply*, *as_reply*, *accepted*)

- Partimos de uno de los casos de prueba originales. El primero que vamos a considerar es el caso:

  (1500, false, low, true)

  En este caso, tenemos una cantidad (*req_amount*) inferior al umbral y por tanto, entraríamos por la rama izquierda. Llamaríamos al asesor (*as_reply*), y en este caso nos indica que el riesgo es bajo. Por tanto, la respuesta del aprobador (*ap_reply*) no interfiere en el resultado final, que en este caso es positivo (*accepted* = *true*).
  Pues bien, para obtener un nuevo caso de prueba a partir de él, vamos a aplicar la siguiente MR en la cual aquellos datos que tienen un 1 son los del caso de prueba original y los datos que tengan un 2 son de un nuevo dato generado (esta notación se sigue también en las siguientes MR descritas):

  Precondición: $req\_amount1 * 10 > 10000$

  MR1: $req\_amount2 = req\_amount1 * 10 \ \wedge \ ap\_reply2 = not(ap\_reply1) \wedge as\_reply2 = as\_reply1 \implies accepted2 = accepted1 \ \wedge \ ap\_reply2 = accepted2$

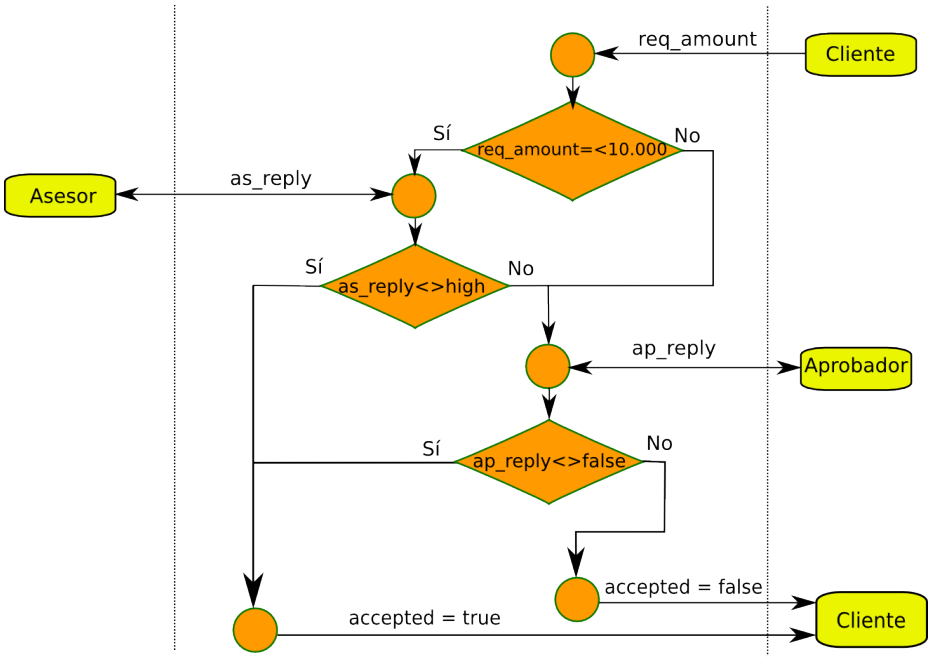


Figura 3: Diagrama de flujo de la composición del Préstamo

La precondición de esta relación es que la cantidad (*req_amount*) multiplicada por 10 sea mayor que el límite. Como este caso de prueba cumple la restricción, al aplicar la MR obtenemos el siguiente caso de prueba:

(15000, true, low, true)

- Ahora, vamos a estudiar un segundo caso de prueba original:

  (1500, false, high, false)

  Para poder obtener un nuevo caso de prueba a partir de él, aplicamos la siguiente precondición y la siguiente relación:

  Precondición: $req\_amount1 * 10 > 10000$

  MR2: $req\_amount2 = req\_amount1 * 10 \ \wedge \ ap\_reply2 = not(ap\_reply1) \wedge as\_reply2 = as\_reply1 \implies accepted2 = not(accepted1) \ \wedge \ ap\_reply2 = accepted2$

  Al aplicar la MR obtenemos el siguiente caso de prueba:

  (15000, true, high, true)

- Por último, vamos a estudiar un tercer caso de prueba que es el siguiente:

(150, false, low, true)

Ahora, para poder obtener un nuevo caso de prueba a partir de él, aplicamos la siguiente precondición y la siguiente relación:

Precondición: $req\_amount1 * 10 \leq 10000$

MR3: $req\_amount2 = req\_amount1 * 10 \ \wedge \ ap\_reply2 = ap\_reply1 \wedge as\_reply2 = as\_reply1 \implies accepted2 = accepted1$

Por tanto, el nuevo caso de prueba que obtenemos es el siguiente:

(1500, false, low, true)

Como podemos ver, éste es el caso original que estudiamos en el primer ejemplo, lo que nos indica que realmente todo este proceso nos puede llevar a nuevos casos de prueba, pero, también a casos de prueba ya estudiados.

5.1. Automatización de la generación de casos de prueba siguientes

Hemos implementado las MR explicadas anteriormente junto a otras que también hemos obtenido, de forma que con ellas realizamos los cambios numéricos y lógicos correspondientes en los valores que poseen los casos de prueba originales y obtenemos los nuevos con los que se mejora el conjunto de casos de prueba inicial.

Como ya hemos comentado en la sección anterior, el estudio de obtención de nuevos casos de prueba nos puede llevar a casos de prueba que ya teníamos anteriormente. Para evitar esta redundancia hemos desarrollado una aplicación que nos permite automatizar la búsqueda de nuevos casos de prueba a partir de otros originales, incluyendo la eliminación de posibles casos de prueba ya existentes en un fichero con formato de BPELUnit.

Para poder llevar a cabo la automatización de la generación de casos de prueba se ha requerido de un estudio exhaustivo de la lógica de la composición y de las posibles propiedades que puedan relacionar los casos de prueba originales con otros nuevos casos.

Pero este esfuerzo se ve recompensado porque esta aplicación nos genera nuevos casos válidos, ahorrándonos el tedioso trabajo de generar nuevos casos de prueba a mano. De hecho, podemos realizar iteraciones, al ejecutar de nuevo la aplicación a todos estos nuevos casos que obtenemos, y así tener un nuevo conjunto de casos de prueba aún más completo.

5.2. Evaluación de resultados

Como resultado global de todo este proceso podemos decir que hemos llegado a generar nuevos casos de prueba automáticamente. Es decir, hemos mejorado la técnica de detección de errores completando el conjunto de datos de prueba que teníamos en un principio con los cuales no se detectaban algunos que sí hemos detectado ahora. Un caso de prueba en particular es el siguiente:

(1500, true, low, true)

Y el caso siguiente generado con la MR2:

(15000, false, low, false)

Pues bien, dada la composición original del Préstamo, cuya lógica interna se puede observar en la figura 3, adjuntamos el fragmento donde se realiza la evaluación de la cantidad requerida (*req_amount*):

```
<condition>
 (number(string($processInput.input/ns0:amount)) <= 10000)
</condition>
```

Si se produce un error en la composición, que consiste en añadir un 0 más a la cantidad umbral (10000), apareciendo 100000, resulta el siguiente fragmento de una composición errónea:

```
<condition>
 (number(string($processInput.input/ns0:amount)) <= 100000)
</condition>
```

Con el caso de prueba original, no se detecta el error en la composición porque sigue cumpliéndose la condición, pero si se observa el caso siguiente generado, donde la cantidad requerida es 15000, podemos comprobar que en la composición original, la condición no se cumple, por lo que se invoca al aprobador y la respuesta depende de éste, mientras que en la composición errónea sigue cumpliéndose la condición al ser 15000 menor que 100000 y se invoca al asesor, que cuando el riesgo es bajo (low) debe aprobar el préstamo (accept= true). Sin embargo, no coincide con el valor esperado (accept=false), por lo que el error es detectado.

Este nuevo caso de prueba es capaz de detectar un error que su caso de prueba original no detectaba.

Por tanto, además de generar un número importante de nuevos casos de prueba de forma automática, algunos de esos casos han detectado nuevos errores, mejorando la cobertura del conjunto de casos de prueba. Los resultados se pueden mejorar también implementando nuevas MR que permitan generar nuevos casos de prueba diferentes de las MR ya implementadas.

6. Trabajos Relacionados

En esta sección se mencionan algunas herramientas y técnicas utilizadas para probar este tipo de software. Nos centraremos en aquellas implementadas para WS-BPEL, especialmente las que estén automatizadas.

La mayoría de las referenciadas en la bibliografía se centran en la generación de casos de prueba para este tipo de composiciones [10,25,27]. Aunque hay trabajos más completos que incluyen la medida de la calidad de los casos de prueba

y la optimización de la generación de los casos de prueba a través de prueba de mutaciones, ninguna se aplica a composiciones de servicios web con WS-BPEL.

GAmera [13] es un entorno de pruebas para composiciones WS-BPEL basado en prueba de mutaciones que utiliza algoritmos genéticos para reducir el número de mutantes requeridos.

El primer trabajo que se conoce relacionado con MT se lo debemos a Weyuker [24]. En él se propone una nueva perspectiva para la prueba de software basada en el uso de programas alternativos al original para probarlo, pero que realizan la misma función (similar a la *prueba por referencia*). Esta idea fue más tarde utilizada por Chen en [6], donde se introduce por primera vez el término de pruebas metamórficas. Desde entonces, se han publicado numerosos trabajos con diferentes enfoques como [12], donde se presenta una posible automatización de esta técnica.

Chen y sus colaboradores en [8] describen la importancia de elegir MR adecuadas al dominio del problema y a la estructura del algoritmo implementado en el programa a probar. Por otro lado, Chan y sus colegas tienen varios trabajos relacionados con este tema, pero uno de los más interesantes para nuestra investigación es [5], pues aplican de MT a arquitecturas orientadas a servicios, aunque no se aplica a composiciones, como en la arquitectura propuesta en este trabajo, sino sólo a servicios. Más recientemente, se ha publicado otro interesante trabajo basado en el análisis de los modelos de características para obtener MR y automatizar la generación de los casos de prueba [23].

Un grupo de la Universidad de Columbia ha implementado en [17] un *framework* que automatiza parte del proceso de prueba. Han analizado programas que implementan máquinas de aprendizaje para automatizar la obtención de propiedades para este tipo de software. Posteriormente han implementado una herramienta llamada Corduroy para automatizar el proceso [18].

7. Conclusiones

Los procesos de negocio basados en WS-BPEL están teniendo gran auge en los últimos años, Por ello, es importante desarrollar técnicas que permitan probar este tipo de software. Al ser un lenguaje propio de los WS es necesario adaptar las técnicas tradicionales de prueba para probar las composiciones basadas en ellos.

Por otro lado, MT ha sido implementada en diferentes lenguajes de manera eficiente y es objeto de estudio e investigación por parte de diversos grupos actualmente. La selección de unas MR adecuadas es un factor crítico en esta técnica, por lo que hay que considerar el contexto del problema y la estructura del programa a probar.

Se ha propuesto una arquitectura de un entorno de pruebas para aplicar MT a las composiciones WS-BPEL. Se especifican sus etapas y el diseño de la misma basándonos en sistemas libres que se han usado en el grupo con anterioridad. También se ha incluido el trabajo realizado sobre un caso de una composición concreta, su especificación, diseño y descripción de la implementación, así como

los resultados obtenidos. Esto muestra cómo es posible mejorar un conjunto de casos de prueba para que detecte errores en la composición.

Como trabajo futuro, está el desarrollo completo de la arquitectura propuesta. Como se ha comentado en el artículo, las piezas básicas están ya implementadas como software libre, por lo que es un trabajo de integración en el que ya se está trabajando con resultados esperanzadores. Una vez esté implementado el sistema, se podría comparar sus resultados con los obtenidos con otras técnicas usando distintas composiciones, aunque lamentablemente no existe hasta la fecha ningúna batería de pruebas estándarizada para WS-BPEL [21], por lo que los resultados podrían ser dependientes de los casos de estudio.

Referencias

1. ActiveVOS: ActiveBPEL WS-BPEL and BPEL4WS engine. http://sourceforge.net/search/?q=ActiveBPEL (Oct 2009)
2. Andrews, J.H., Briand, L.C., Labiche, Y.: Is mutation an appropriate tool for testing experiments? In: Proceedings of the 27th International Conference on Software Engineering (ICSE 2005). pp. 402–411. ACM Press (2005)
3. Beizer, B.: Software Testing Techniques, 2nd Edition. International Thomson Computer Press, 2 sub edn. (Jun 1990)
4. Bozkurt, M., Harman, M., Hassoun, Y.: TR-10-01: testing web services: A survey. Tech. Rep. TR-10-01, King's College, London (2010)
5. Chan, W.K., Cheung, S.C., Leung, K.R.: A metamorphic testing approach for online testing of service-oriented software applications. International Journal of Web Services Research 4(2), 61–81 (2007)
6. Chen, T.Y.: Metamorphic testing: A new approach for generating next test cases. HKUSTCS98-01 (1998)
7. Chen, T.Y.: Metamorphic testing: A simple approach to alleviate the oracle problem. In: Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering. IEEE Computer Society (2010)
8. Chen, T.Y., Huang, D.H., Tse, T.H., Zhou, Z.Q.: Case studies on the selection of useful relations in metamorphic testing. In: Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004). pp. 569–583 (2004)
9. Domínguez Jiménez, J.J., Estero Botaro, A., Medina Bulo, I., Palomo Duarte, M., Palomo Lozano, F.: El reto de los servicios web para el software libre. In: Proceedings of the FLOSS International Conference 2007. pp. 117–132. Servicio de Publicaciones de la Universidad de Cádiz (Mar 2007)
10. García-Fanjul, J., Tuya, J., de la Riva, C.: Generación sistemática de pruebas para composiciones de servicios utilizando criterios de suficiencia basados en transiciones. In: JISBD 2007: Actas de las XII Jornadas de Ingeniería del Software y Bases de Datos (2007)
11. García Domínguez, A., Medina Bulo, I., Marcos Bárcena, M.: Inference of performance constraints in Web Service composition models. In: Proceedings of the 2nd International Workshop on Model-Driven Service Engineering. Málaga, Spain (Jul 2010), `http://www.kybele.etsii.urjc.es/mose2010/`, to be published
12. Gotlieb, A., Botella, B.: Automated metamorphic testing. Computer Software and Applications Conference, Annual International 0, 34–40 (2003)

13. Grupo SPI&FM: GAmera home site. http://neptuno.uca.es/~gamera (2010)
14. IDC: Research reports. http://www.idc.com (2008)
15. Leveson, N.G.: Safeware: system safety and computers. ACM (1995)
16. Mayer, P., Lübke, D.: Towards a BPEL unit testing framework. In: TAV-WEB'06: Proceedings of the 2006 workshop on Testing, Analysis, and Verification of Web Services and Applications. pp. 33–42. ACM (2006)
17. Murphy, C., Kaiser, G., Hu, L., Wu, L.: Properties of machine learning applications for use in metamorphic testing. In: Proc. of the 20th international conference on software engineering and knowledge engineering (SEKE). pp. 867–872 (2008)
18. Murphy, C., Shen, K., Kaiser, G.: Using JML runtime assertion checking to automate metamorphic testing in applications without test oracles. In: Software Testing Verification and Validation, 2009. ICST'09. International Conference on. pp. 436–445 (2009)
19. Myers, G.J., Sandler, C., Badgett, T., Thomas, T. M.: The Art of Software Testing, 2nd ed. Wiley - Interscience (2004)
20. OASIS: Web Services Business Process Execution Language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (2007), Organization for the Advancement of Structured Information Standards
21. Palomo-Duarte, M.: Service composition verification and validation. In: Jonathan Lee, S.P.M., Liu, A. (eds.) Service Life Cycle Tools and Technologies: Methods, Trends and Advances. IGI Global (A la espera de publicación)
22. Palomo-Duarte, M., García-Domínguez, A., Medina-Bulo, I., Álvarez-Ayllón, A., Santacruz, J.: Takuan: a tool for WS-BPEL composition testing using dynamic invariant generation. In: et al., B.B. (ed.) ICWE. Lecture Notes in Computer Science, vol. 6189, pp. 532–535. Springer (2010)
23. Segura, S., Hierons, R., Benavides, D., Ruiz-Cortés, A.: Automated test data generation on the analyses of feature models: A metamorphic testing approach. In: International Conference on Software Testing, Verification and Validation. pp. 35–44. IEEE press (2010)
24. Weyuker, E.: On testing Non-Testable programs. The Computer Journal 25(4), 465–470 (Nov 1982)
25. Yan, J., Li, Z., Yuan, Y., Sun, W., Zhang, J.: BPEL4WS unit testing: Test case generation using a concurrent path analysis approach. In: ISSRE 2006: 17th International Symposium on Software Reliability Engineering. pp. 75–84. IEEE Computer Society (2006)
26. Zhang, Z.Y., Chan, W.K., Tse, T.H., Hu, P.F.: An experimental study to compare the use of metamorphic testing and assertion checking. Journal of Software 20(10), 2637–2654 (2009)
27. Zheng, Y., Zhou, J., Krause, P.: An automatic test case generation framework for web services. Journal of software 2(3), 64–77 (2007)
28. Zhou, Z.Q., Huang, D.H., Tse, T.H., Yang, Z., Huang, H., Chen, T.Y.: Metamorphic testing and its applications. In: Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004). Software Engineers Association (2004)

Gestión de la composición semántica de servicios web para el dominio de turismo

María-Belén Rayo, Manuel Noguera, José Luis Garrido, Kawtar Benghazi

E.T.S.I.I.T., Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada
C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, España
brayol@correo.ugr.es
{mnoguera,jgarrido,benghazi}@ugr.es

Abstract. Un aspecto clave a la hora de desarrollar aplicaciones basadas en servicios consiste en la capacidad que éstas deben poseer para descubrir y componer servicios de forma consistente. Hasta ahora, la implementación de las interconexiones entre servicios viene realizándose manualmente, de forma que los desarrolladores y agentes informáticos han de identificar el significado de parámetros y operaciones involucrados en cada composición a partir de la sintaxis y documentación, con frecuencia difusa, de éstos. El presente trabajo tiene por objeto utilizar un enfoque ontológico que permita descubrir y componer servicios de forma semi-automática a partir de descripciones semánticas. Este objetivo se logra mediante especificaciones ontológicas que permiten el descubrimiento de servicios web, así como definir reglas para su composición con otros servicios. Asimismo, se ha desarrollado una aplicación de usuario para la creación de aplicaciones basadas en servicios y que hace uso de las descripciones semánticas mencionadas. La propuesta se describe mediante un ejemplo perteneciente al dominio del turismo [4].

1 Introducción

La arquitectura orientada a servicios, SOA (Service Oriented Architecture), es un modelo de referencia que propone la utilización de servicios para dar soporte a los requisitos de negocio, brindando una forma bien definida de publicación e invocación de los mismos que facilita la interacción entre diferentes sistemas propios o de terceros [1].

Debido al auge del uso de servicios web como mejor solución estándar de desarrollo de aplicaciones orientadas a servicios, han aparecido numerosos repositorios con información de este tipo de servicios, gracias a los que es posible desarrollar aplicaciones mediante la conexión de servicios desarrollados por terceros. Sin embargo, estas interconexiones se deben hacer, en la mayoría de los casos, de forma manual y con documentación incompleta [2]. Por lo tanto, es necesario disponer de mecanismos que ayuden en la tarea de composición de servicios para resolver problemas concretos.

El problema de la composición de servicios web ha sido abarcado tradicionalmente desde el punto de vista sintáctico, desarrollando algoritmos que, basándose en los

tipos de datos y nombres de los parámetros, tratan de detectar cuándo dos servicios son "componibles". Sin embargo, estas soluciones resultan ineficaces debido a la ambigüedad del lenguaje y a la falta de mecanismos que medien entre dos servicios, independientemente de que sintácticamente sean similares o no. Con idea de mejorar los procesos de composición de servicios, se ha abordado el tema haciendo uso de tecnologías para la Web Semántica, surgiendo de este modo los servicios web semánticos, considerados como una nueva tecnología resultante de la combinación de la Web Semántica y los servicios web [22]. El objetivo es aprovechar la fuerza y mejorar las limitaciones de ambas tecnologías por separado. En este contexto, aparece el reto de diseñar una herramienta capaz de componer servicios a través de descripciones semánticas que hacen uso de una ontología de conceptos acerca de servicios, de forma que los parámetros de los servicios se anotan con respecto a los conceptos de la misma.

El presente trabajo muestra el desarrollo de una aplicación, acotada por un dominio, en este caso, el turismo, que permita llevar a cabo y dé soporte a la composición semántica de servicios web. Esta propuesta parte inicialmente de un modelo ontológico que define las relaciones que se tienen que establecer entre una serie de servicios para que estos puedan componerse. A partir de aquí, junto con los servicios concretos que se quieran componer y las preferencias de usuario. A partir de ahí, se efectúa un razonamiento que permite deducir la composición [10] de servicios más adecuada. Esta propuesta persigue el objetivo, no sólo de aumentar la interoperabilidad entre aplicaciones, sino poder ofrecer al usuario una herramienta que tenga en cuenta sus preferencias.

El resto del artículo contiene cuatro secciones adicionales. La sección 2 repasa las principales propuestas que existen hasta ahora para la composición y descubrimiento de servicios, principalmente en el dominio de los servicios web. La sección 3 presenta tecnologías basadas en servicios web, donde se describe la pila de protocolos estándares más utilizados actualmente y se ilustra la idea de composición semántica de servicios. La sección 4 muestra un caso de estudio donde se describe una ontología para descubrir servicios y, por último, la sección 5 expone las conclusiones que se derivan del trabajo realizado.

2 Trabajos relacionados

Hasta ahora existen diversos enfoques para la descripción y anotación semántica de servicios web, entre los cuales destacan OWL-S y WSMO, de los que hablaremos a continuación. Sin embargo, contrastando con lo anterior escasean propuestas que permitan la composición semántica de servicios, principalmente en el ámbito de los servicios web.

OWL-S [19] se basa en la definición de varias ontologías escritas en OWL para describir la semántica de las propiedades funcionales y no funcionales de un servicio, así como la interacción entre el cliente y el proveedor del servicio. Según este enfoque, la anotación semántica permite plasmar qué ofrece el servicio, cómo funciona y cómo se debe interactuar con él, empleando para ello tres secciones: profile, model y grounding.

La primera sección sería la encargada de responder a la pregunta ¿qué ofrece el servicio?, haciendo uso para ello de una ontología llamada Service Profile en la que se definen propiedades del servicio; como su nombre, su descripción y toda aquella información necesaria para poder identificarlo. Además se describen las entradas, las salidas, las precondiciones y postcondiciones que se deben cumplir antes y después de su ejecución, así como otras propiedades adicionales.

La segunda sección se encarga de responder a la pregunta ¿cómo funciona el servicio?, y lo hace a través de una ontología llamada Service Model. En este enfoque cualquier servicio es visto como un proceso (atómico o compuesto) que proporciona una salida que cumple ciertas precondiciones, para unos parámetros de entrada. Los parámetros de entrada y salida son representados como variables de SWRL (Service Web Rule Language) y en cada uno de ellos se define el tipo de valores que puede instanciar. La transferencia de datos entre procesos se formaliza como flujos datos y será necesario que todos los procesos terminen para que se dé por finalizada la sentencia que los agrupa y así poder asegurar la obtención del resultado deseado.

Por último la ontología Service Grounding es la encargada de definir la correspondencia entre la descripción del servicio y la descripción semántica. Especifica cómo se interactúa con el servicio, describiendo el formato de los mensajes, el protocolo que lo implementa, el transporte y todo lo necesario para completar la descripción semántica.

Sin embargo OWL-S [14] sólo permite describir servicios web semánticos y carece de mecanismos que permitan componer servicios, ya que OWL – S no soporta orquestación.

Por otro lado, WSMO (Web Service Modeling Ontology) [7] es una ontología para servicios web, basada en el *framework* WSMF, que no sólo realiza una descripción semántica del servicio, sino que además añade una descripción de todos los elementos que componen la infraestructura necesaria para realizar la invocación, descubrimiento y composición de un servicio. Además formaliza los objetivos del cliente a la hora de invocar el servicio y los mediadores que son los encargados de enlazar todos los componentes de la infraestructura, resolviendo los desajustes que pueda ocasionar el poder tener servicios distribuidos y así garantizar la interoperabilidad entre ellos. Para dar soporte a WSMO, se han desarrollado implementaciones de referencia como WSMX o IRS-III.

IRS-III [3] está basado en IRS-II y es una plataforma que actúa como agente intermediario entre las peticiones del cliente y los servicios web disponibles. Sigue los principios de diseño de WSMO y su arquitectura esta diseña en tres capas: Servidor, publicador y cliente. El servidor está basado en un servidor http y es el encargado de proporcionar soporte al protocolo soap. El publicador es el encargado de asociar un servicio web con una descripción del servicios basada en WSMO y el cliente que proporciona un mecanismo basado en objetivos para realizar la invocación de los servicios. La composición de servicios web que ofrece este enfoque, consiste en considerar la unidad básica del modelo un objetivo. Dicho objetivo se puede separar en un conjunto de metas, creando así un flujo de datos que será manejado por IRS-III. Para conseguir cada una de estas metas, es invocado el servicio web adecuado y el resultado será devuelto, de forma que no hay interacción directa entre los servicios involucrados. De manera que a pesar de que sea posible la orquestación de servicios,

carece de mecanismos que permitan componer los servicios identificados e involucrados y que se pueda producir interacción directa entre ellos.

Otro posible enfoque para la composición de servicios es Ontobroker [17]; un gestor para la composición semántica de servicios que apoya todas las recomendaciones proporcionadas por W3C para la web semántica. Soporta OWL, RDF, RDFS, SPARQL, además de la industria estándar F-Logic. Es capaz de resolver consultas complejas en pequeñas cantidades de tiempo gracias a sus arquitecturas distribuidas. Ofrece una interfaz de servicios web y la capacidad de expresar en lenguaje natural como se han obtenido las respuestas en función de la información dada. Ontobroker actúa como motor de inferencia para procesar ontologías, de forma que será necesario definir una ontología que muestre como deben ser los parámetros de entrada, de salida y las condiciones necesarias que deben darse para que dos servicios se puedan relacionar y así poder realizar la composición. Aún así, una vez que han sido identificados los servicios, no tiene definidos mecanismos para la construcción de aplicaciones, ni queda reflejado como se realiza la orquestación de dos servicios en una descripción formal que pueda ser interpretada por alguna herramienta, para poder crear aplicaciones a partir de la definición de dicha composición de servicios.

Como bien se puede observar en esta revisión, se ha estudiado y avanzado sobre la incorporación de la semántica al mundo de los servicios web. Se han desarrollado mecanismos para describir a los servicios desde el punto de vista semántico y así poder realizar un descubrimiento que se adecúe a las necesidades del usuario. También se ha avanzado sobre la composición de servicios [12], creando herramientas capaces de identificar servicios relacionados e interoperables, pero sin embargo, a pesar de los esfuerzos realizados hasta ahora, carece la existencia de técnicas con soporte tecnológico que permitan crear aplicaciones una vez que dichos servicios son identificados, reto a conseguir por la propuesta plasmada en el presente trabajo.

3 Tecnologías para el descubrimiento y composición de servicios web

El escenario común para el funcionamiento de servicios web se construye gracias a la interacción de tres componentes software [21]: proveedor del servicio, encargado de alojar al servicio y proporcionar su interfaz; registro del servicio, encargado de mantener la información del servicio e intervenir en los procesos de búsqueda y la aplicación cliente, cuyos requerimientos pretenden satisfacerse con la funcionalidad ofrecida por el servicio. Para garantizar la interoperabilidad y a la vez hacer posible una combinación de servicios que permita resolver operaciones más complejas, es necesario construir este escenario sobre una arquitectura de referencia creada sobre un conjunto de estándares establecidos.

Procesos de Orquestación. **BPEL**
Descubrimiento. **UDDI**
Descripción. **WSDL**
Intercambio de Mensajes. **SOAP**
Transferencia de Mensajes. **HTTP**

Figura1: Arquitectura de tecnología de servicios web

Transferencia de mensajes. HTTP: La capa más baja de la pila es la encargada de realizar la transferencia de mensajes a través de protocolos como HTTP (Hypertext Transfer Protocol) o SMTP (Simple Mail Transfer Protocol) y de definir las direcciones únicas de los recursos.

Intercambio de mensajes. SOAP: "Soap (Simple Object Access Protocol) es un protocolo estándar basado en XML que define como dos objetos en diferentes procesos pueden interaccionar por medio de un intercambio de mensajes". [16]

Tanto las peticiones de los clientes, como las respuestas de los servicios, suelen realizarse a través de mensajes, todos ellos ejecutándose en diferentes plataformas; por lo tanto, será en la siguiente capa de la pila donde el protocolo SOAP se encargue de estandarizar cómo deben ser esos mensajes para que el intercambio sea correcto. Las características que diferencian a SOAP [8] de otros protocolos y lo hacen más adecuado para estandarizar el intercambio de mensajes en la tecnología de servicios web, son el hecho de no estar asociado a un solo lenguaje o una sola plataforma, que no se encuentra fuertemente asociado a ningún protocolo de transportes, que aprovecha los estándares ya existentes y permite la interoperabilidad entre múltiples entornos.

Descripción del servicio web. WSDL: Cuando una aplicación ofrece alguna funcionalidad a través de ciertas operaciones, es necesario proporcionarle información para que las operaciones se completen y la aplicación devuelva los resultados al cliente. Estos intercambios de información se pueden llevar a cabo gracias a una descripción de las interfaces de acceso al servicio. En el protocolo SOAP no hay una forma estándar de hacer especificaciones, es por ello por lo que IBM y Microsoft han propuesto el lenguaje de descripción de servicios WSDL (Web Services Description Language) [11] que es un documento XML compuesto por una parte abstracta donde se definen los tipos, los mensajes y las operaciones, y una parte concreta donde se define el protocolo de transporte y otras informaciones.

Las principales ventajas que ofrece WSDL son que facilita escribir y mantener servicios mediante una definición de interfaces web, facilita el acceso a los servicios y facilita hacer cambios para ampliar los servicios. Así, un programa cliente que se conecta a un servicio web, puede leer el documento WSDL y determinar que funciones están disponibles en el servidor, facilitando el trabajo a los desarrolladores de aplicaciones clientes.

Descubrimiento del servicio. UDDI: El descubrimiento de servicios Web [6] es un proceso que consiste en localizar, uno o varios documentos relacionados que describen un servicio web determinado mediante el lenguaje de descripción de servicios Web (WSDL), ofreciendo a los clientes la posibilidad de conocer fácilmente la existencia del servicio y la ubicación del documento que lo describe.

En la actualidad existe un mecanismo llamado UDDI (Universal Description Discovery and Integration) [9] que es un registro público diseñado para almacenar información acerca de las empresas y los servicios que éstas ofrecen. Está basado en XML y es un estándar básico de los servicios web cuyo objetivo es ser accedido por lo mensajes SOAP y dar paso a documentos WSDL. Sirve como infraestructura para una colección de software basado en servicios web, donde se pueden realizar búsquedas sistemáticas y categóricas de los datos utilizando sistemas basados en taxonomías, de forma universal e independiente de los proveedores.

Orquestación e interacción de servicios: Una vez que se han descubierto y se conoce la existencia de los servicios, llegamos a la capa más alta, que será la encargada de establecer la interacción entre ellos. En ella se define procesos de negocio, de descubrimiento, de agregación así como la orquestación que consiste en conectar servicios web entre sí para crear procesos de negocio de alto nivel y conseguir un resultado derivado de la asociación correcta de las operaciones. La especificación más conocida de procesos de negocio, [1] es WS-BPEL; un lenguaje de orquestación estandarizado por OASIS, basado en XML y diseñado para controlar la invocación de diferentes servicios web, con cierta lógica de negocio; ayudando así a la programación a gran escala.

4 Compositor semántico de servicios web para el dominio turístico

La tecnología de servicios web formada por un conjunto de componentes software distribuidos, relacionados e interoperables, expuesta en el apartado anterior, ha supuesto un avance en la programación de aplicaciones. Sin embargo WSDL sólo proporciona una descripción de la funcionalidad de los servicios a nivel sintáctico, careciendo de descripciones semánticas para interpretar el significado de los parámetros y mensajes. A medida que se profundiza en el desarrollo y aplicaciones SOA, y aumenta su diversidad y complejidad [5], resulta necesario añadir información que permita hacer afirmaciones sobre los parámetros y operaciones, pudiendo así dar a conocer el tipo, relaciones de herencia y restricciones de cardinalidad, además de toda aquella información adicional necesaria para que un razonador sea capaz, a partir de una descripción semántica del servicio que buscamos y una lista de descripciones de servicios existentes, de seleccionar el servicio adecuado de forma automática [15].

La propuesta que presentamos en este trabajo consiste en realizar una aplicación, que a partir de unos parámetros de entrada, sea capaz de descubrir servicios relacionados en el dominio del turismo y pueda componerlos para conseguir el objetivo final que pretende satisfacer el cliente. Para ello vamos a trabajar con un conjunto de servicios web que están relacionados semánticamente, como son reserva

de vuelos, alquiler de vehículos y reserva de hotel; de forma que los parámetros de salida de un servicio pueden ser también parámetros de entrada de otro servicio, en función, no sólo de la correspondencia entre los tipos de los parámetros, sino también de otros elementos del contexto en el que se provee el servicio, tales como la localización, el perfil del consumidor (negocios, familiar, económico) o sus características personales (edad, capacidades/discapacidades, intereses), etc.

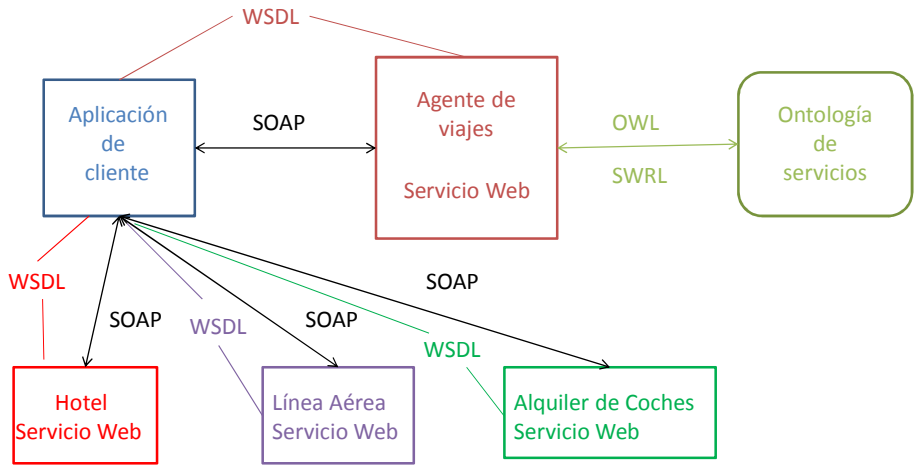


Figura 2: Diseño esquemático de la propuesta

4.1 Arquitectura del sistema

El término ontología hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades. [18]

La arquitectura del sistema está compuesta principalmente por una ontología para descubrir los servicios a la que se le puede preguntar, en términos generales, qué servicios se podrían componer. Esta primera ontología nos podría proporcionar aquellos servicios que pertenecen al mismo dominio, contexto en el que se provee el servicio, y que "*a priori*" podrían estar relacionados. Las descripciones acerca de los servicios de dicha ontología son utilizadas por un Subsistema de Interpretación que procesa las especificaciones y ofrece al usuario diseñador de servicios una interfaz para componerlos. Una vez que se define la composición, ésta queda reflejada en otra ontología de orquestación, donde además se especifican otros parámetros que pudiera introducir el usuario final y que permitiese mejorar el resultado de la ejecución de los servicios. Esta Ontología de Orquestación será interpretada por un Subsistema de Generación de Interfaces de Usuario; que presentará a los usuarios finales del sistema una interfaz de aplicación que les permita interactuar con el servicio obtenido como resultado de la composición.

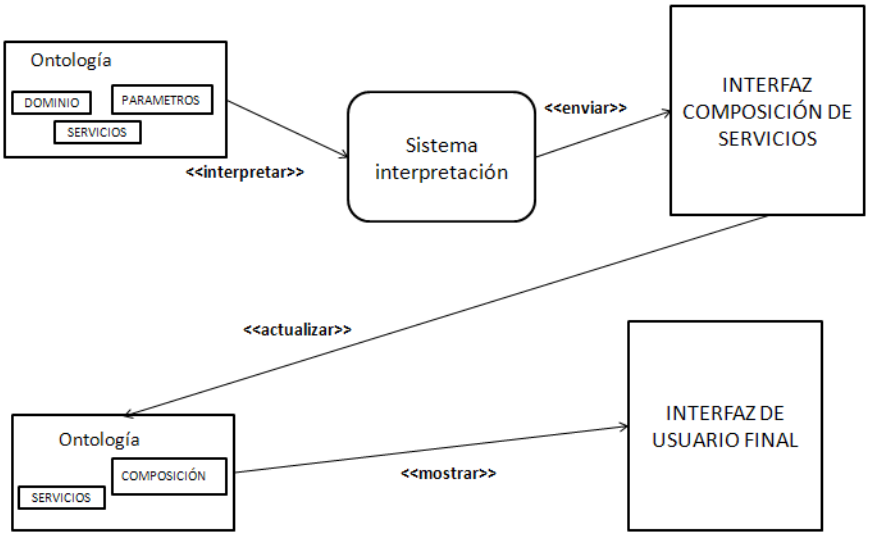


Figura 3 : Arquitectura del sistema

4.2 Ontologías de descripción de servicios

La ontología de descripción de servicios es el componente principal de la arquitectura del sistema y será la encargada de aportar las descripciones semánticas sobre las que se desarrollará la aplicación final. Esta ontología está compuesta por tres clases principales: Dominio, Parámetros y Servicios; donde Dominio marca el ámbito en el que se provee el servicio; Parámetros, fomada por dos subclases, representa los parámetros de entrada y salida; y Servicio, que describe al propio servicio y tendrá tantas subclases como servicios disponibles que en este caso específico serían 3 (alquiler de coches, reserva de hotel y reserva de vuelos).

Una vez definidas las clases, la ontología debe presentar unas propiedades que las relacionen. Cada Servicio tendrá un dominio y unos parámetros, de forma que existirá una relación entre Servicio y Dominio, y entre Servicio y Parámetros, además de las relaciones inversas. Posteriormente se crearán tantas instancias para cada clase como objetos del mundo real se quieran tratar en la ontología.

Con este esquema general, la ontología marcará la base de conocimiento para que un razonador sea capaz de agrupar servicios que se puedan componer en función de su dominio y de sus parámetros.

Para definir la implementación de la ontología hemos utilizado la herramienta *Protegé* [23], que también nos ha permitido reflejarla en el formato gráfico mostrado a continuación.

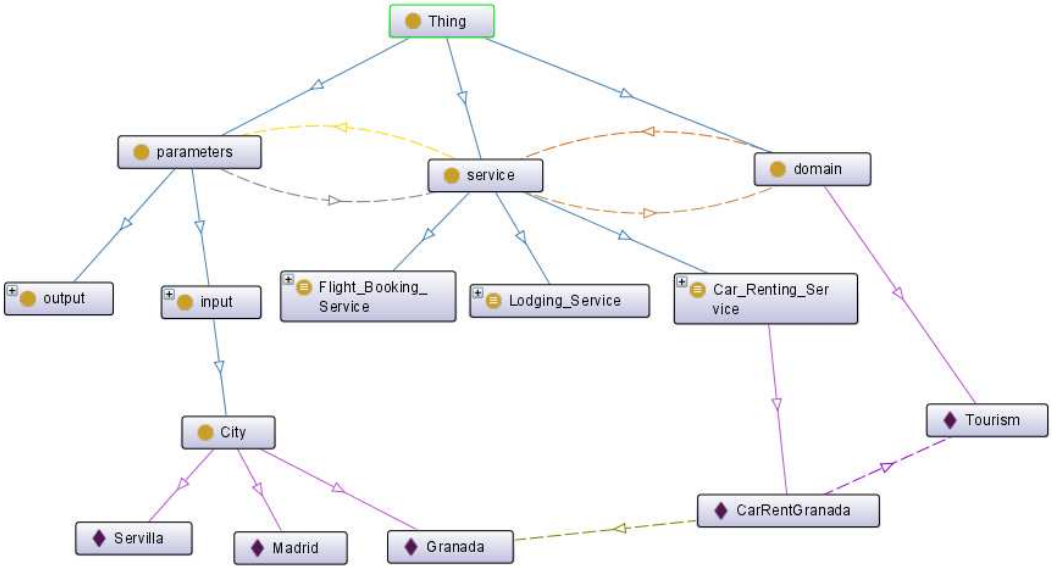


Figura 4: Ontología que describe servicios web

4.3 Sistema de Interpretación.

Una vez definida la ontología de orquestación, la herramienta protegé [23] la almacena en un fichero .owl [24] y será la API Protegé – OWL [25] la que proporcione los mecanismos necesarios para interpretarla y manipularla.

Protegé- OWL API [25] es una biblioteca de JAVA para Ontology Web Language (OWL) que proporciona clases y métodos para cargar y guardar archivos OWL, consultar y manipular los datos de OWL, y para llevar a cabo el razonamiento. De esta forma, a través de un entorno de desarrollo para el lenguaje JAVA como NetBeans [27], y añadiéndole esta biblioteca, interpretaremos la ontología.

SWRL (Semantic Web Rule Language) [26] es un lenguaje que resulta de la combinación de los sublenguajes OWL DL, OWL Lite y RuleML, que permite la creación de reglas mediante una sintaxis abstracta de alto nivel que amplía la de OWL [24]. Una vez que tenemos cargada la ontología mediante Protege-OWL [25], emplearemos SWRL para crear las consultas necesarias que nos permitan ir obteniendo como consecuentes aquellos servicios con los que se podría componer el servicio antecedente que le proporcionamos a la regla y así poder ofrecer al usuario una composición de los mismos en función de la semántica establecida por la ontología.

En la figura 5 se muestra una regla que selecciona aquellos servicios pertenecientes al dominio del turismo, aplicada sobre la ontología definida anteriormente. Esta regla es la primera que la aplicación de usuario utiliza para componer aplicaciones basadas en servicios de un dominio concreto.

```
"PREFIX po: http://www.semanticweb.org/ontologies/2011/2/services.owl#
"+ "SELECT ?services "+ "WHERE  "+ "{  "+ "po:Tourism  po : isDomainOf  ?services . "+"}";
```

Figura 5: Regla SWRL para seleccionar servicios pertenecientes al dominio del turismo

4.4 Interfaz de Composición de Servicios

Una vez interpretada la ontología, el siguiente paso que toma protagonismo en la arquitectura del sistema es el diseño de la interfaz, mostrando al usuario diseñador los resultados obtenidos, es decir, los servicios descubiertos y para cada uno de ellos, aquellos con los que sería posible definir una composición. La idea general que presentamos en esta parte de la propuesta es la de plasmar en un primer panel los servicios disponibles, y una vez seleccionado un servicio, mostrar en un segundo panel, los servicios con los que se podría componer. De esta forma quedarían reflejadas las posibles composiciones semánticas de los servicios que se podrían realizar. Para ello hemos creado un servicio web que proporcione un listado de los servicios pertenecientes al dominio del Turismo en la ontología y se creará posteriormente una aplicación web cliente que lo consuma, de forma que pueda establecer la relación entre ellos y ofrecer una posible composición.

En la figura 4 el diseño de la interfaz de aplicación web cliente que pretendemos desarrollar para mostrar la composición de los servicios al usuario. Se ha seleccionado el servicio Flight_Booking_Service a modo de ejemplo, y aparecen Lodging_Service, Car_Renting_Service y Visit_Booking_Service como servicios disponibles con los que se podría componer.

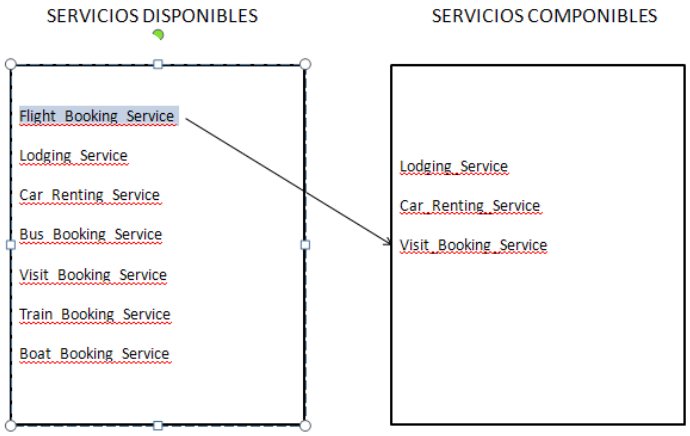


Figura 6: Interfaz de composición de servicios

Posteriormente dicha composición quedará recogida en una segunda ontología, que desarrollaremos en un futuro y conformará un segundo recurso del sistema, con información semántica a modo de base para la generación de la interfaz de usuario final.

4.5 Diseño de Aplicaciones Finales

El último componente que cierra la arquitectura del sistema es el diseño de la interfaz final de la aplicación, con la que el usuario establecerá una comunicación. Este diseño estará limitado por el dominio en el que nos encontramos durante toda la propuesta y pretendemos mostrar una interfaz dinámica y atractiva que permita una interacción sencilla con el usuario.

La principal tarea que se pretende llevar a cabo a través de este diseño es la de recoger de forma exhaustiva todas las preferencias del usuario para poder descubrir y componer los servicios de forma adecuada. Para ello vamos a tomar como servicio principal el de reserva de vuelos y, por lo tanto, se pedirá al usuario toda la información necesaria que formará los parámetros de entrada para invocar al servicio. Posteriormente, una vez que se ha seleccionado el servicio adecuado, se mostrará al usuario los parámetros de salida y aquellos servicios con los que se podría componer, en nuestro caso un servicio de alquiler de vehículos y otro de reservas de alojamiento. Este nuevo servicio también requerirá unos parámetros, pero sólo serán solicitados aquellos que no sean resultado del servicio de la reserva de vuelos, ya que al componerse tomará como parámetros de entrada los parámetros de salida o de entrada del servicio anterior. El objetivo es ir proporcionando sucesivamente otros servicios con los que componer los ya seleccionados conforme se va realizando la composición, permitiendo ofrecer al usuario una aplicación dinámica e interoperable desarrollada sobre el ámbito de los servicios web.

Finalmente, una vez que se ha finalizado la composición, la aplicación mostrará una pantalla final al usuario donde quedará reflejado el resultado final con los servicios solicitados.

5 Conclusiones

Las tecnologías de los servicios web tradicionales realmente cumplen con el objetivo de aumentar la interoperabilidad entre aplicaciones, pero aún presentan carencias para descubrir, invocar y componer servicios lo que le resta potencia a los sistemas basados en el paradigma SOA, principalmente del ámbito de los servicios web. Para intentar solventar estos inconvenientes, se ha presentado una propuesta que hace uso de tecnologías. En particular se propone una arquitectura de sistema, que hace uso de ontologías para describir servicios semánticamente, así como su composición a través de anotaciones semánticas que permita construir aplicaciones basadas en servicios web para el dominio turístico, que permita realizar las tareas de descubrir, invocar y componer de forma semi- automática.

Las anotaciones semánticas principales aportadas se rigen a través de una ontología y consiste en asociar conceptos y relaciones de la misma con los parámetros y operaciones de un servicio web.

A través de la descripción explícita acerca de los servicios mediante ontologías, sus parámetros, su dominio, se puede razonar sobre cómo componerlos y construir aplicaciones que puedan interpretar dichas ontologías. Ello, además, permite automatizar el proceso y mejorarlo, permitiendo así, hacer la composición de forma consistente a partir de la descripción semántica de los servicios. Finalmente, esto también hace más independientes a los analistas y diseñadores de las aplicaciones, ya que podemos cambiar la ontología, sin necesidad de tener que cambiar la aplicación.

En el futuro pretendemos desarrollar posibles mejoras a la propuesta, añadiendo servicios web que proporcionen mayor funcionalidad dentro del dominio del turismo; como puede ser la reserva de visitas, ofertar la elección de transporte con el que realizar el viaje (no sólo los vuelos), y ampliar la aplicación a otros dominios que le aporten una envergadura más amplia, compleja y acorde a las; ya tan avanzadas necesidades de los usuarios en la actualidad.

Agradecimientos

Este trabajo de investigación ha sido financiado por los proyectos TIN 2007-60199 y TIN2008-05995/TSI del Ministerio de Ciencia e Innovación.

Referencias

1. De la Rosa, J.L., Triviño, A., Aldana J.F., Servicios Semánticos : OWL-S y WSMO. Proyecto Fin de Carrera de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Málaga.
2. Rao, J., Xiaomeng, S., A Survey of Automated Web Service Composition Methods. Norwegian University of Science and Technology.
3. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B,, Pedrinaci, C., IRS-II: A bróker-based approach to semantic Web Services. Journal of Web Semantics ScienceDirect. March 2008
4. Kopecky, J., Simperl, E., Semantic Web Service Offer Discovery with Lightweight. Semantic Technology Institute (STI Innsbruck). Innsbruck, Austria.
5. Guzmán, J.A., Modelo de Planificación y Ejecución Concurrente para la Composición de Servicios Web Semánticos en Entornos Parcialmente Observables. Tesis Doctoral de la Universidad Nacional de Colombia. Diciembre 2009.
6. Atos Origin.Proyecto GODO: Generación inteligente de Objetivos para el Descubrimiento de servicios web semánticos. Octubre 2007.
7. Proyecto PLATA (Plataforma de Libre Acceso para Tecnologías Avanzadas en la Web).Laboratorio de Televisión Digital Interactiva. Universidad de Vigo.2007
8. Guía Breve de Servicios Web.World Wide Web Consortium. W3C. Oficina Española. http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb

9. Manual de Servicios Web en plataforma.NET. http://www.desarrolloweb.com/articulos/1589.php
10. Giraldo, J., Guzmán, J., Ledesma, A., Sistema Multiagente para la composición de servicios web semánticos.Universidad Nacional de Colombia. Noviembre 2007.
11. How to publish and find WSDL service descriptions. http://www.ibm.com/developerworks/webservices/library/ws-wsdl/
12. Álvarez, P.J, Bañares J.A., Conceptos Básicos de coordinación de servicios web. Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza.
13. Ortiz, P., Riestra, L., Descubrimiento de servicios para la evolución dinámica de sistemas software mediante transformación de modelos.
14. Bechhofer, S., Programming to the OWL API. University of Manchester.
15. Sungin, L., Senator, J., Hong-Gee., K., Hanmin, J., Mikyoung, L., Seungjae, S., Beom-Jong, Y., OntoPipeliner: An ontology-based automatic semantic service pipeline generator. Expert Systems with Applications. 2011.
16. SOAP.Wikipedia. http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol
17. BERROCAL, J.L., et al. Agentes inteligentes: recuperación autónoma de información en la web. Revista Española de Documentación Científica, 2003, vol. 26, nº 1
18. ONTOLOGIA Wikipedia . http://es.wikipedia.org/wiki/Ontolog%C3%ADa_(inform%C3%A1tica)
19. OWL-S Coalition. Noviembre 2004. OWL-S:Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/
20. Apache Software Foundation. 2008. jUDDI. http://juddi.apache.org/
21. Booth, D., Haas, H. y otros. Febrero 2004. Web Services Architecture. http://www.w3.org/TR/ws-arch/
22. Cabral, L., Dominique, J., Motta, E., Aproaches to Semantic Web Services: An overview and Comparisons.
23. *Protegé*. Open source ontology editor and knowledge-base framework. http://protege.stanford.edu/.
24. OWL Web Ontology Language. http://www.w3.org/TR/owl-features/.
25. Protegé-OWL API. http://protege.stanford.edu/plugins/owl/api/.
26. Semantic Web Rule Lenguaje. SWRL. http://www.w3.org/Submission/SWRL/
27. NetBeans. http://netbeans.org/

Generación automática de esqueletos de mecanismos de integración del ESB, basada en herramientas semánticas

Juan José Herrera Martín[1] y Joatham Pérez Expósito[1]

[1] Novasoft Soluciones Canarias S.A.,
Santa Cruz de Tenerife, España
{jhmartin, jpexposito}@novasoft.es
http://www.novasoft.es

Resumen

El ESB [1] (Enterprise Service Bus) es un elemento clave en el diseño de plataformas de Interoperabilidad basadas en SOA [2] (Service Oriented Architecture), para la implementación de mecanismos de integración y orquestación de servicios. En este artículo se presenta una aproximación para la generación de esqueletos de estos mecanismos a través de herramientas semánticas, en base a las experiencias obtenidas en el proyecto PLATINO [3] (Plataforma de Interoperabilidad del Gobierno de Canarias).

1. Introducción

El Bus de Servicios Empresarial, más conocido por sus siglas en inglés ESB (Enterprise Service Bus), es una plataforma de software que actúa como capa de integración intermedia (middleware) entre distintos sistemas o aplicaciones, por medio de servicios de comunicación y transformación de mensajes basada en estándares. La construcción de estos servicios, se realiza combinando los distintos componentes que ofrece el ESB que implementan funcionalidades de tratamiento de mensajes, tales como enrutamiento, transformación, orquestación, etc.

Gracias a sus características para soportar múltiples paradigmas de integración así como para centralizar el control y distribuir el procesamiento, el ESB se ha convertido en un elemento clave en el diseño de plataformas de interoperabilidad basadas en SOA (Service Oriented Architecture). Un ejemplo de ello, lo podemos observar en el proyecto PLATINO (Plataforma de Interoperabilidad del Gobierno de Canarias) , el cual ofrece a los distintos sistemas backoffice y frontoffice (sede electrónicas, portales, etc) del Gobierno de Canarias, un conjunto de servicios corporativos haciendo uso de estas prácticas.

Actualmente la plataforma PLATINO ofrece un conjunto de 20 servicios corporativos, los cuales presentan gran heterogeneidad en el diseño de sus mecanismos de integración en el ESB. En este trabajo se presenta una aproximación para la generación automática de esqueletos de los mecanismos de integración en el ESB, en base a las características de servicio y haciendo uso de herramientas semánticas, surgida desde el proyecto PLATINO con la intención de facilitar la incorporación y mantenimiento de nuevos servicios a la plataforma y normalizar su diseño en base a unos patrones definidos.

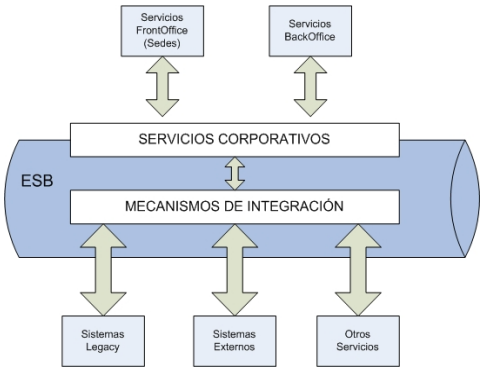


Fig. 1. *Arquitectura general de PLATINO*

2. Descripción general del sistema

En la siguiente figura se presenta un esquema general de la aproximación propuesta.

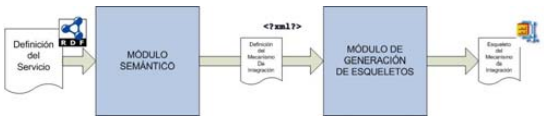


Fig. 2. *Esquema general del sistema*

Inicialmente se parte de la definición del servicio que se va a incorporar a la plataforma, la cual se envía al módulo semántico.
El módulo semántico es el encargado de validar y almacenar la definición del servicio. Posteriormente, infiere el tipo de mecanismo de integración de ESB a implementar para el servicio. Y finalmente recopila la información necesaria para la generación de su esqueleto.
Por último el módulo de generación de esqueletos desarrolla el esqueleto del mecanismo de integración de

ESB, a partir de la información obtenida del módulo semántico.
En los siguientes apartados se describe con mayor detalle cada una de estas fases.

3. Definición del servicio

La definición del nuevo servicio de la plataforma se realiza por medio de un documento RDF (Resource Description Framework) [4], en el cual se incluye información del servicio sobre:

- Descripción general del servicio.
- Protocolo de comunicación del servicio.
- Definición de los métodos del servicio.
    - Descripción del método.
    - Parámetros de entrada y salida del método.
    - Servicios invocados por el método.

4. Módulo semántico

El módulo semántico es el encargado de interpretar la definición del servicio e inferir el tipo de mecanismo de integración de ESB a utilizar. Para la realización de estas tareas, el módulo semántico incluye una ontología OWL [5] y conjunto de reglas que modelan el conocimiento asociado a los servicios y a los mecanismos de integración del ESB.

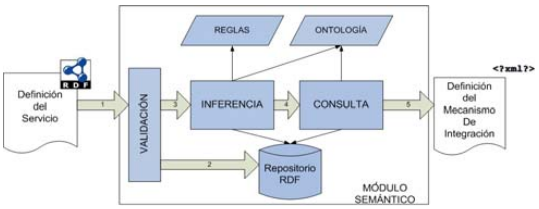


Fig. 3. *Esquema del módulo semántico*

Como se muestra en el esquema de la figura 3, en primer lugar el módulo semántico recibe la definición de servicio y por medio del componente de validación se comprueba el formato del documento RDF y se almacena en el repositorio. A continuación el componente de inferencia obtiene el tipo de mecanismo de integración de ESB a utilizar para la implementación del servicio, a partir de la ontología, las reglas y la definición del servicio. Finalmente el componente de consulta, se encarga de recopilar la información necesaria y construir la definición de esqueleto del tipo de mecanismo de integración inferido.

Destacar que para la implementación del módulo semántico se utilizó la plataforma Jena. [6]

5. Generación de esqueletos

Este módulo es el encargado de desarrollar los esqueletos de los mecanismos de integración de ESB indicado por el módulo semántico para el nuevo servicio. Para la implementación de este módulo se ha utilizado a ServiceMix [7] como ESB de referencia, ya que es el utilizado en el proyecto PLATINO.

Las tareas del módulo de generación de esqueletos se pueden dividir en las siguientes fases:

- Validación del fichero XML de definición de esqueleto.
- Generación de los arquetipos de los distintos componentes de ServiceMix (binding components y services engine) que conforman el mecanismo de integración.
- Configuración de los componentes de ServiceMix. En esta fase se configurarían los distintos ficheros xbean.xml y el resto de ficheros de configuración de los componentes.
- Generación de los esqueletos de las clases POJO y otras unidades de software que requieran los componentes de ServiceMix implicados en el mecanismo de integración.
- Creación del ServiceAssembly que se desplegará en ServiceMix, que será el producto final del módulo.

Para la implementación de este módulo se ha utilizado Java y Maven. [8]

6. Conclusiones y trabajos futuros

A raíz de la experiencia obtenida en el proyecto PLATINO y en el desarrollo de esta aproximación podemos destacar los siguientes aspectos:

- La normalización de los mecanismos de integración de ESB, favorece el mantenimiento de plataformas de interoperabilidad como el proyecto PLATINO.
- El uso de herramientas semánticas para la definición de los servicios, ofrece un gran potencial para la gestión y consulta de su información asociada.

Por otro lado, las futuras líneas de trabajos se centran en el estudio e incorporación a la aproximación de nuevos modelos de mecanismos de integración de ESB tomando como referencia el ServiceMix.

7. Agradecimientos

Destacar y agradecer su colaboración para la elaboración de este articulo a la Dirección General de Telecomunicaciones y Nuevas Tecnologías (DGTNT) del Gobierno de Canarias a la Oficina Técnica del proyecto PLATINO y a la empresa Novasoft.

Referencias

[1] Chappell, David A: Enterprise Service Bus, Theory in Practice o'Reilly (2004)

[2] Thomas, Erl.: Service-Oriented Architecture: Concept, Technology, and Design. Prentice Hall (2005)

[3] PLATINO, http://www.gobiernodecanarias.org/platino

[4] Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 Febrero 2004

[5] OWL Web Ontology Language Overview. W3C. Recommendation 10 Febrero 2004.

[6] Jena, http://jena.sourceforge.net/

[7] ServiceMix, http://servicemix.apache.org/

[8] Maven, http://maven.apache.org/

Sesión 2: Análisis y optimización en BPM

COMBAS: una herramienta para el análisis de procesos con información semántica*

E. González-López de Murillas, J. Fabra, P. Álvarez y J. Ezpeleta

Instituto de Investigación en Ingeniería de Aragón (I3A)
Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza
María de Luna 3, E-50018 Zaragoza (España)
{edugonza,jfabra,alvaper,ezpeleta}@unizar.es

Resumen Desde la concepción de un sistema hasta su puesta en funcionamiento de una manera fiable hay un largo camino. Existen numerosas herramientas y técnicas que facilitan la tarea. En el contexto actual, la adición de información semántica en la especificación, análisis e implementación de sistema basados en tecnologías Web hace que las técnicas y tecnologías utilizadas deban ser adaptadas/extendidas para considerar estos aspectos.
En este trabajo se presenta COMBAS (COmprobador de Modelos BAsado en Semántica), un entorno que integra herramientas para la verificación de modelos (*model checking*) de procesos que incluyen información semántica. COMBAS permite, a partir de un modelo descrito mediante una clase de Redes de Petri de alto nivel anotadas con información semántica, las U-RDF-PN, la generación de su grafo de alcanzabilidad, la edición de fórmulas en lógica temporal para la expresión de las propiedades que se desean verificar en el modelo y la visualización de resultados, incluyendo la posibilidad de navegar por el grafo de estados alcanzables y visualizar sus anotaciones. Como caso de estudio se presenta la aplicación de estas herramientas al *First Provenance Challenge*, un caso de *workflow* tomado del mundo de la computación científica.

Palabras clave: Procesos con Anotaciones Semánticas, RDF, Redes de Petri, *Model Checking*.

1. Introducción

Para automatizar el ciclo de vida de los procesos de negocio es necesaria la descripción semántica de los diferentes elementos que los constituyen (tareas, estructuras y condiciones de control, datos, mensajes intercambiados entre procesos, etc.). Estos procesos descritos semánticamente reciben el nombre de *Procesos de Negocio Semánticos* (PNS). Hasta la fecha, los principales esfuerzos se han centrado en definir lenguajes de descripción semántica (RDF, WSMO u

* Este trabajo se ha financiado parcialmente gracias al proyecto de investigación del Ministerio de Ciencia e Innovación TIN2010-17905.

OWL, entre otros) y ontologías del dominio de aplicación, y en integrar estas soluciones en los lenguajes de procesos de negocio más utilizados (WS-BPEL, por ejemplo). Una vez disponemos de soluciones para modelar e incluso ejecutar PNS, el siguiente paso natural debería ser proponer soluciones que permitan analizar cómo se van a comportar estos procesos en ejecución, considerando como parte del análisis las descripciones semánticas de los mismos. En este sentido, se deberán plantear la utilización de nuevas versiones de las técnicas tradicionales de análisis con este nuevo tipo de procesos que, además, se están extendiendo ampliamente al campo de los *workflows* científicos, dada la utilidad del análisis aplicado a problemas cuya ejecución requiere un elevado coste en tiempo e infraestructura.

En [1] los autores presentaron un método de análisis basado en *Model Checking* para procesos de negocio semánticos. Este método permitía predecir el comportamiento de procesos con descripciones RDF utilizando lógica CTL (*Computation Tree Logic*) también enriquecida con información semántica. Las principales contribuciones del método eran: i) el análisis consideraba aspectos de control y datos (tanto internos del proceso, como los intercambiados con otros procesos) del proceso; y ii) no sólo se consideraban los datos conocidos en tiempo de diseño, sino también los que serán instanciados en tiempo de ejecución (datos simbólicos). Como parte del trabajo, finalmente, también se presentaba una implementación del método aplicado a un caso concreto y que justificaba la viabilidad del mismo.

En este trabajo se propone una generalización del prototipo anterior y una extensión y mejora. Esto se ha conseguido mediante la adición de una serie de herramientas y técnicas que facilitan las diferentes etapas necesarias para la verificación del modelo correspondiente. Como resultado presentamos COMBAS, un entorno que integra las herramientas necesarias para realizar *model checking* sobre PNS anotados mediante RDF y utilizando CTL. COMBAS abarca todas las fases del análisis, permitiendo la generación de los grafos de alcanzabilidad a partir del *workflow* de un PNS descrito mediante la herramienta Renew [2] y redes de Petri de alto nivel [1], el procesamiento de las anotaciones semánticas, la edición y visualización de fórmulas en CTL, la verificación del modelo y la posterior visualización gráfica de los resultados. COMBAS representa un enfoque totalmente novedoso frente a la ausencia y limitación de herramientas y *frameworks* para abordar los problemas de procesos con información semántica.

Ya existen numerosas herramientas de *model checking* que pueden clasificarse en función del lenguaje de modelado y de propiedades que utilicen. De entre todas ellas, vamos a centrarnos en las principales y que aceptan CTL como lenguaje de expresión de propiedades[1]. Algunas herramientas permiten definir propiedades en CTL y LTL. BANDERA [3] utiliza la técnica de análisis de código para verificar propiedades sobre modelos definidos en Java, y CADENCE SMV [3] usa una técnica de model checking plano sobre modelos expresados en los lenguajes Cadence SMV, SMV o Verilog. Otro ejemplo es NuSMV, que permite, además

[1] En `http://anna.fi.muni.cz/yahoda/` se puede encontrar una extensa comparativa de estas herramientas.

de CTL y LTL, definir propiedades en lenguaje PSL para hacer model checking plano sobre modelos SMV. APMC [4] usa una técnica aproximada y probabilística sobre *Reactive Modules* y las propiedades se describen en los lenguajes PCTL y PLTL. PRISM también soporta ambos lenguajes, además de CSL, y usa una técnica probabilística sobre modelos en gran variedad de lenguajes como PEPA, PRISM language o Plain MC [4]. Entre los *model checker* probabilísticos también encontramos MCMR, que soporta propiedades en lenguajes CSL, CSRL, PCTL o PRCTL y hace la verificación usando técnicas de tiempo real y probabilísticas sobre modelos modelados en Plain MC.

Otras herramientas aceptan propiedades en el lenguaje π-calculus, como TAPAs que también soporta CTL sobre modelos CCSP, y ARC que también acepta propiedades en CTL* haciendo model checking plano sobre modelos AltaRica. GEAR es otra herramienta que también acepta propiedades en π-calculus, además de en AFMC y CTL [5]. Similar a esta última encontramos CWB-NC que, usando model checking plano y temporizado sobre modelos CCS, CSP, LOTOS y TCCS, soporta propiedades en lenguajes AFMC, CTL y GCTL. Por último, existe una aplicación, MCMAS, que hace model checking plano y epistémico sobre modelos ISPL verificando propiedades en lenguajes CTL y CTLK [6].

Sin embargo, no existe ninguna herramienta de *model checking* que soporte la inclusión de información semántica RDF en el modelo de origen, su procesamiento y su posterior análisis. Esto justifica la utilidad de COMBAS, representando una alternativa muy válida para abordar este tipo de problemas.

Este artículo está organizado de la siguiente manera. La Sección 2 presenta el diseño de COMBAS y algunos detalles de implementación. Su aplicabilidad se demuestra en la Sección 3 mediante un caso de estudio. Finalmente, se presentan las conclusiones.

2. COMBAS: COmprobador de Modelos BAsado en Semántica

La herramienta desarrollada, llamada COMBAS (*COmprobador de Modelos BAsado en Semántica*), posibilita la verificación formal de un sistema que se le pasa como entrada mediante la utilización de bases de datos semánticas, un algoritmo de *Model Checking* que soporta RDF y predicados descritos en CTL. Como resultado, COMBAS determina si el modelo verifica o no las propiedades.

La herramienta cubre cinco funcionalidades: la generación de grafos de alcanzabilidad, revisión de grafos, edición de fórmulas CTL, *CTL Model Checking* y la revisión de resultados.

En la Figura 1 se muestra la arquitectura del sistema. El proceso es como sigue:

1. El ingeniero diseña el *workflow* con la herramienta gráfica Renew, guardándolo como PNML. Debe generar también los ficheros XML con la descripción de grafos y patrones que etiquetan el modelo.

2. El generador de grafos de alcanzabilidad se alimenta de los ficheros anteriores, produciendo dos salidas. Una es el grafo de alcanzabilidad almacenado en el *RDF Triple Store*. La otra salida está formada por un fichero XML que contiene la relación entre estados y su marcado, una colección de ficheros XML con los RDFGs (*RDF Graph Pattern*) correspondientes a los marcados y un fichero *dot* con la representación gráfica del grafo de alcanzabilidad, visible con la herramienta *Graphviz*.
3. El visor Web utilizará los ficheros XML y gráficos generados mediante el generador de grafos para visualizar el resultado.
4. Se debe crear la fórmula CTL a verificar con el *model checker*. Esta fórmula puede generarse usando el editor Web, que la exportará como un fichero XML.
5. El *Model checker* utiliza la fórmula CTL descrita en XML y generada con el editor, así como el grafo de alcanzabilidad almacenado en el *Triple Store*, y genera como salida una colección de ficheros XML relacionados que representan los estados del grafo de alcanzabilidad que verifican las partes correspondientes de la fórmula CTL. Dichos ficheros permiten la visualización y análisis en la interfaz Web.

2.1. Generación del grafo de alcanzabilidad

La generación del grafo de alcanzabilidad se realiza a partir del modelo de entrada: una red de Petri de la clase de las U-RDF-PN [1] con un marcado inicial, correspondiente al estado inicial del sistema. Las anotaciones semánticas pueden aparecer sobre tres tipos de elementos diferentes: 1) en los arcos, donde pueden aparecer patrones RDF; 2) en las transiciones, para definir las guardas asociadas así como las postcondiciones, y 3) en los lugares, donde los *tokens* también serán grafos RDF.

El grafo de alcanzabilidad generado también se almacena como tripletas RDF, según la ontología que se muestra en la Figura 2.

La generación del grafo de alcanzabilidad se ha implementado mediante la adaptación a la naturaleza semántica de las anotaciones del algoritmo clásico, tal y como se esquematiza a continuación:

1. Transformación del fuente PNML con la U-RDF-PN en RDF, de acuerdo a la ontología en la Figura 3, y almacenamiento en el *triple store*;
2. Introducción del marcado inicial m_0 en la pila de 'Marcados por procesar' y el conjunto de 'Estados' alcanzables (inicialmente vacío);
3. Mientras que ('Marcados por procesar' != vacía) {
   a) m_i = desapilado de la cima de 'No revisados';
   b) Obtención de las transiciones sensibilizadas por el marcado m_i (con sus *bindings* correspondientes) ;
   c) Generación de los marcados resultantes del disparo de dichas transiciones;
   d) Para cada m_j generado en el paso anterior {

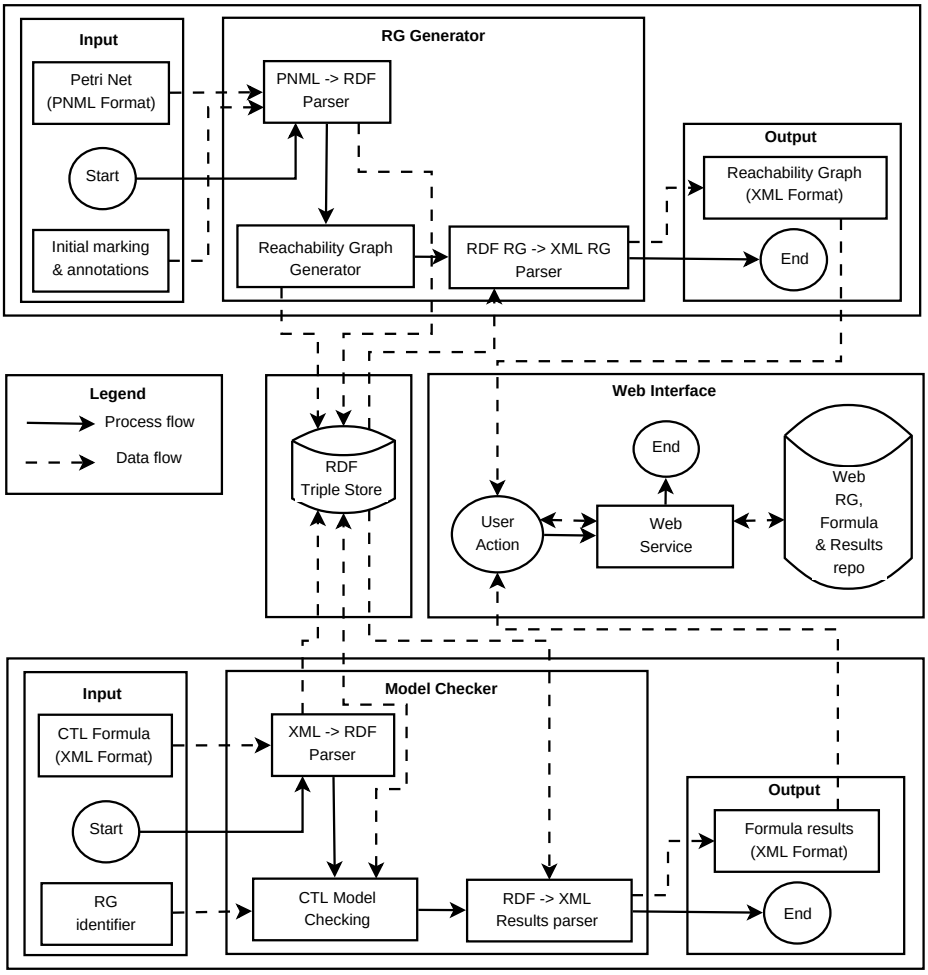


Figura 1. Arquitectura de COMBAS

1) Si no existe en 'Estados' uno equivalente a m_j, añadir m_j a 'Estados' y apilarlo en 'Marcados por procesar';
2) Añadir transición (m_i, m_j) al grafo; }

4. } Si no se generó ningún marcado nuevo en el paso anterior, enlazar el estado m_i consigo mismo, para convertirlo en un estado de *deadlock* o final (requisito para que el resultado sea una *estructura de Kripke*, necesaria como entrada al *model checker*).

Una vez obtenida la representación RDF del grafo de alcanzabilidad (*Reachability Graph*, RG), se genera el XML correspondiente.

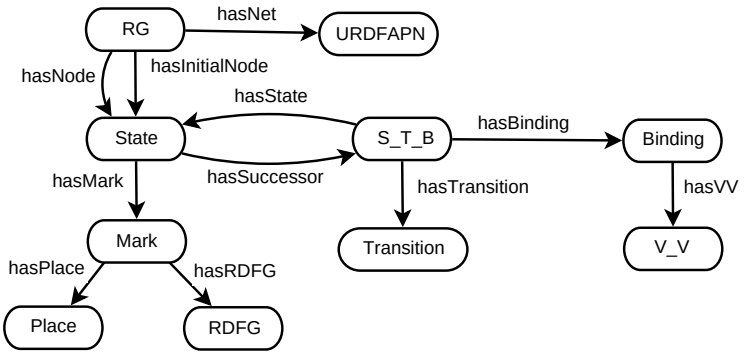


Figura 2. Diagrama de la ontología para definir el grafo de alcanzabilidad

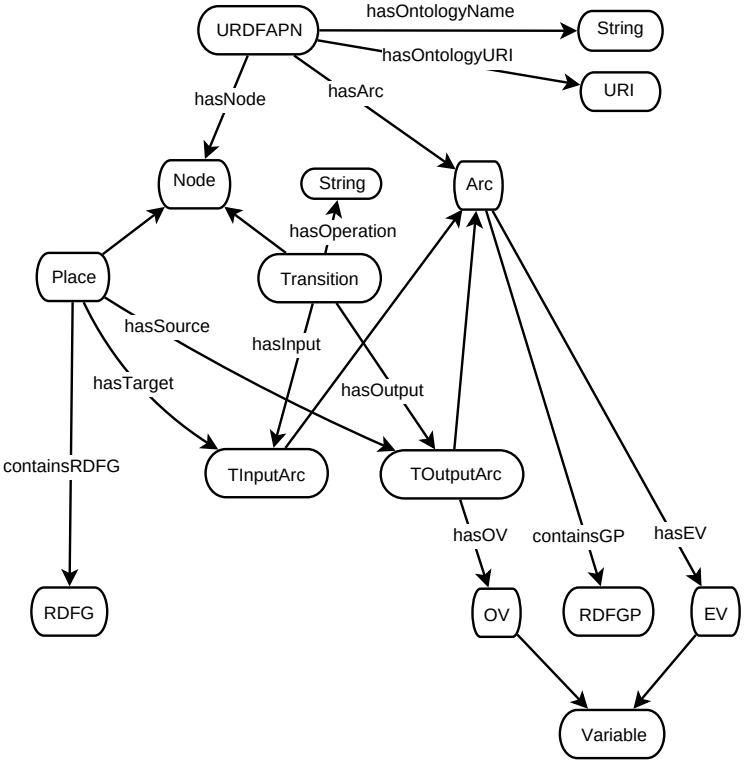


Figura 3. Diagrama de la ontología para describir U-RDF-PN

2.2. Servicio de edición y visualización

Las funciones básicas de la interfaz Web de usuario son la visualización de los grafos de alcanzabilidad generados por el *RG-Generator*, la construcción y edición de las fórmulas CTL que usará el *Model Checker* y la revisión de los

resultados de la verificación de propiedades. Aquí, la idea principal fue integrar todas estas funciones en un mismo interfaz, de forma que interactúen entre sí. Pasemos, por tanto, a describir con algo más de detalle cada una de ellas.

Visualización de grafos de alcanzabilidad. Como se ha explicado anteriormente, la salida del generador está formada por un fichero principal que describe la estructura del grafo de alcanzabilidad y una colección de ficheros que contienen los RDFG del marcado de cada estado. Todos estos ficheros están descritos en XML. El grafo generado puede contener un número de estados muy grande, del orden de cientos en un caso simple, pero puede crecer exponencialmente conforme aumenta la complejidad. Así, la revisión de la corrección del grafo puede ser una ardua tarea. De ahí surgió la idea de implementar un visor, que represente el grafo mediante un diagrama y permita consultar el marcado de cada estado.

Con la aplicación desarrollada es posible seleccionar un grafo de alcanzabilidad determinado, visualizar su estructura en forma de diagrama y consultar el marcado de cada estado.

Edición de fórmulas. Contar con una herramienta de edición de fórmulas CTL eficaz beneficia el proceso de verificación, minimizando errores en escritura, mejorando la experiencia visual y colaborando, en definitiva, a reducir tanto la curva de aprendizaje por parte del usuario como los tiempos de creación y edición de fórmulas. Por todo esto, es importante diseñar una forma intuitiva de interactuar con el usuario. Como idea inicial pareció acertado desarrollar un constructor/editor interactivo de fórmulas mediante una lista de componentes que funcionan como *nodos* de un árbol. A su vez, se consideró interesante añadir mecanismos que garantizan que la fórmula construida es válida y no contiene errores sintácticos. Además, se permite la visualización de la fórmula de forma gráfica en todo momento.

Revisión de los resultados del *Model Checker*. Tan importante como la edición de fórmulas es la revisión de los resultados la evaluación respecto a un modelo determinado. Saber si el modelo satisface la fórmula o no es inmediato, puesto que el *Model Checker* lo especifica al finalizar su ejecución. Sin embargo, el análisis de los resultados de la verificación puede ser tedioso si no se cuenta con una herramienta que facilite el proceso. Este caso es especialmente importante si el modelo no satisface la fórmula, ya que entonces es necesario analizar en qué estados del modelo se evalúa a falsa. Por esta razón, se decidió implementar una herramienta integrada en el visor de grafos que permite inspeccionar el resultado del análisis en un estado concreto y, a su vez, el marcado de dicho estado.

De esta forma, es posible seleccionar un grafo y observar rápida e intuitivamente el resultado de la ejecución de una fórmula, así como los estados del grafo, consultando qué nodos de la fórmula satisface cada estado.

2.3. Representación de la fórmula CTL mediante RDF

La fórmula CTL es uno de los parámetros de entrada del *model checker*, por lo que es necesario definir un formato de representación válido para alimentar la herramienta. La fórmula se describe mediante RDF de acuerdo con la ontología mostrada en la Figura 4 y, como se ha dicho, se almacena como un fichero en XML.

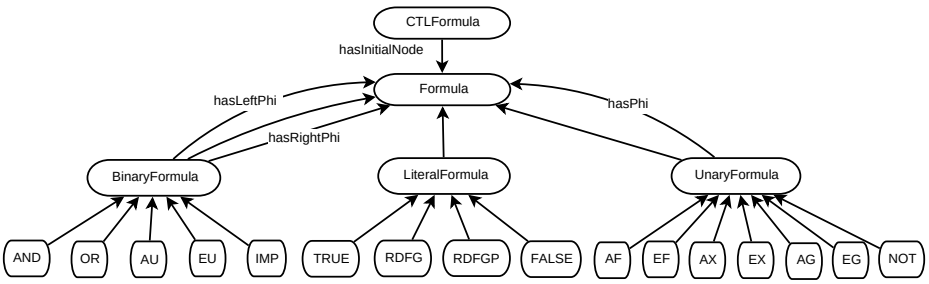


Figura 4. Diagrama de la ontología diseñada para definir fórmulas CTL

3. El *First Provenance Challenge*

Para evaluar el sistema se consideró interesante utilizar un modelo de *workflow* científico. Un buen ejemplo es el caso del *First Provenance Challenge* [7]. Se trata de un problema de referencia en *workflows* científicos por su complejidad y escalabilidad, y se puede definir y modelar con redes de Petri. La posibilidad de escalar la complejidad del problema posibilita que el grafo de alcanzabilidad generado crezca exponencialmente, de manera que el espacio de estados en los que se verifican las propiedades también aumenta su complejidad.

El *First Provenance Challenge* propone un ejemplo de *workflow* científico para crear *atlas cerebrales* de la población, obteniendo los datos del archivo de datos anatómicos en alta resolución del *fMRI Data Center* [8]. El *workflow* se esquematiza en la Figura 5, donde se pueden diferenciar cinco fases. Los procedimientos utilizados en cada etapa hacen uso del juego de herramientas AIR (*Automated Image Registration*) [9] para crear una imagen cerebral media a partir de una colección de datos anatómicos en alta resolución, y el juego de herramientas FSL [10] para crear imágenes 2D de las capas en cada dimensión del cerebro. Además de los datos representados en el diagrama, existen una serie de parámetros (constantes) que se proporcionan a los procesos, y que especifican opciones de configuración [7].

Los parámetros de entrada del *workflow* son un conjunto de imágenes cerebrales (*Anatomy Images*) y una imagen cerebral de referencia (*Reference Image*). Todas estas imágenes son escaneos 3D de un cerebro en varias resoluciones, de

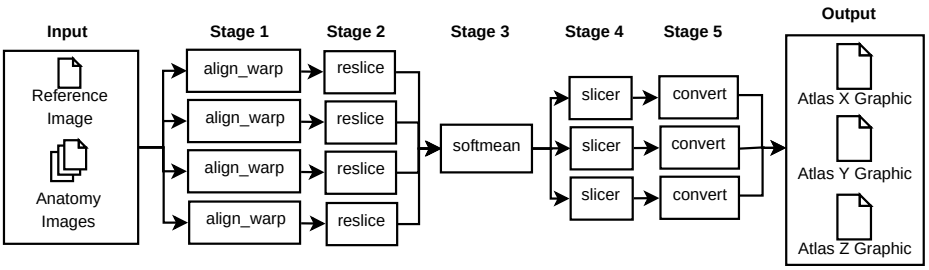


Figura 5. Diagrama del *workflow* correspondiente al *First Provenance Challenge*

modo que aparecen diferencias evidentes. Para cada imagen, además de la propia imagen existe información adicional. Los datos de imágenes fueron publicados en [11].

En la primera etapa, la herramienta *align_warp* compara la imagen de referencia con cada imagen cerebral para determinar cómo debe ser tratada (posición y forma de la imagen ajustada) para encajar con la imagen cerebral de referencia. La salida de cada procedimiento en esta fase es un conjunto de parámetros *warp* que definen la transformación espacial que se realizará. La transformación de la imagen se realiza en la fase *reslice* a partir de cada conjunto de parámetros *warp*, generando una nueva versión de la imagen original como salida. A continuación, todas las imágenes transformadas son promediadas en una sola usando un proceso de *softmean* en la tercera etapa. Para cada dimensión (*x, y, z*), la imagen promedio se trocea para obtener un mapa 2D del plano desde el centro de la imagen en 3D. La salida es un *atlas* de imágenes generado mediante *slicer*, una herramienta que forma parte del juego de utilidades FSL [10]. Finalmente, cada conjunto de datos es transformado en una imagen utilizando *convert*, incluido en la utilidad GNU *ImageMagick*.

La U-RDF-PN que modela el *workflow* se muestra en la Figura 6. A partir de este modelo se genera el grafo de alcanzabilidad, cuya versión simplificada (debido a su gran tamaño) se muestra en la Figura 7. Como se ha comentado, la generación del grafo de alcanzabilidad es una tarea muy costosa debido, fundamentalmente, a la ejecución de dos operaciones: la que determina los posibles *bindings* para la sensibilización de una transición y, la más costosa, la que determina si un potencial estado nuevo es equivalente a uno ya calculado anteriormente. Ambas operaciones hacen un uso intensivo de invocaciones a operaciones del *RDF-store*. El coste real obtenido para la generación del RG utilizando uno o dos juegos distintos de imágenes se muestra en la Tabla 1, donde se aprecia claramente el aumento del coste computacional al escalar el problema. Las ejecuciones se realizaron sobre un procesador Intel Core 2 Duo E7400 de 64 bits (2 x 2.8GHz), con 4GB de RAM, sistema de E/S SATA y corriendo un SO Ubuntu 11.04 con un kernel optimizado 2.6.38.8 y la JVM JRE6u24. El *RDF-store* utilizado fue la versión 6.1.1 de Virtuoso Open Source.

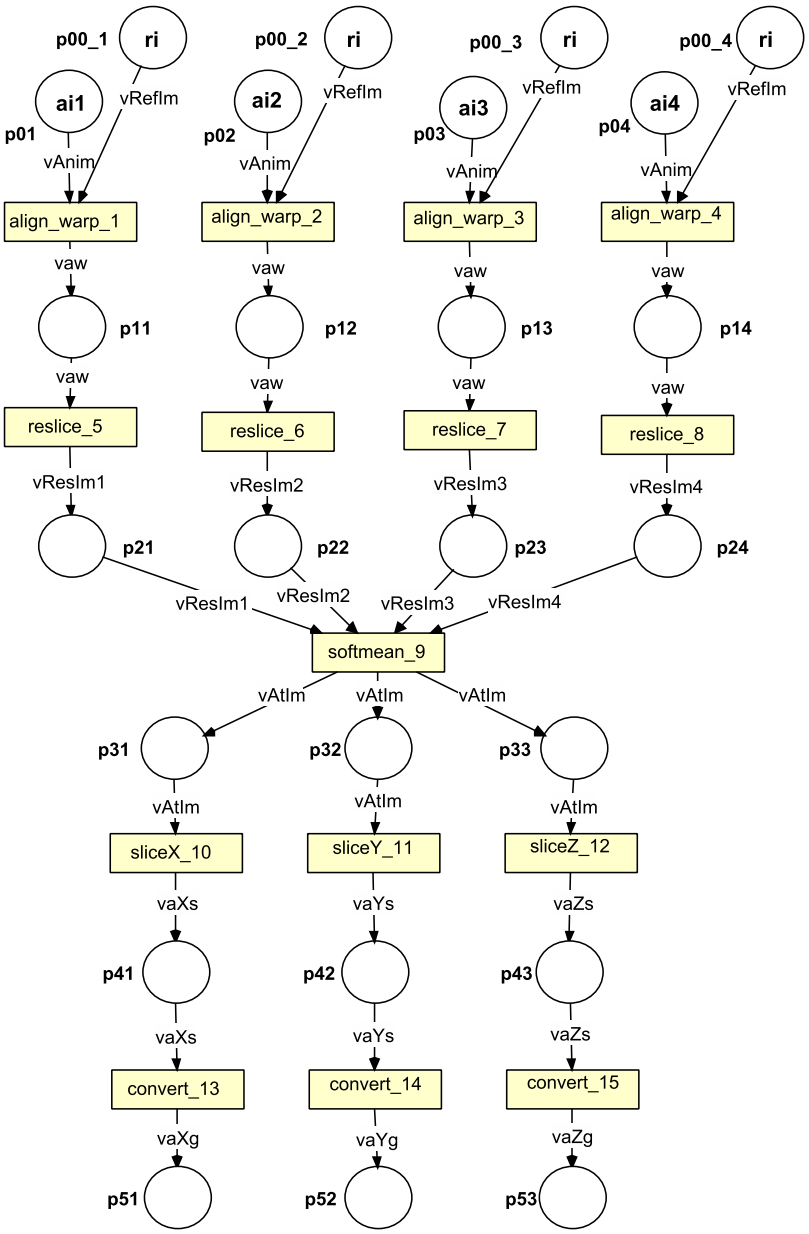


Figura 6. U-RDF-PN correspondiente al *First Provenance Challenge*

3.1. Ejecución y resultados de *Model Checking*

Una vez generado el grafo de alcanzabilidad, se pasa a la fase de *model checking*. Para ello es necesario establecer cuáles son las fórmulas que representan

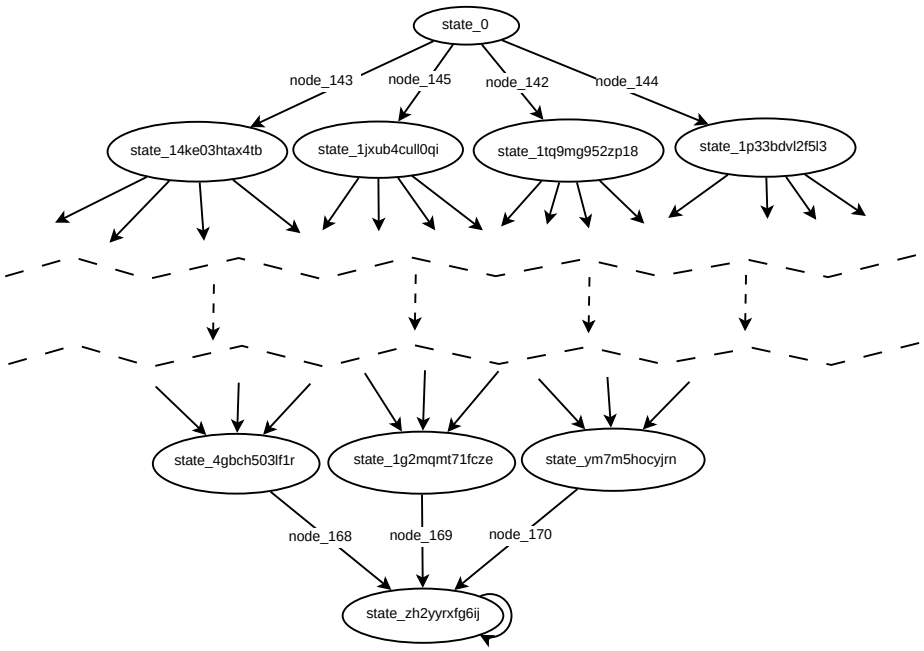


Figura 7. Diagrama resumido del grafo generado para el *First Provenance Challenge*

Caso	Tiempo de generación	Estados generados
1 paquete de imágenes	2 min 16 seg	108
2 paquetes de imágenes	44h y 23 min	11664

Tabla 1. Resultados de la ejecución del generador de RG para el ejemplo

las propiedades que queremos probar. Nuestro modelo corresponde a un sistema que, a partir de un conjunto de cinco imágenes cerebrales, genera un promedio de cuatro de ellas y extrae las vistas correspondientes a las dimensiones X, Y y Z del cerebro. De este modo, al final, obtenemos tres imágenes en 2D codificadas en formato GIF. Además, para garantizar el procesado por paquetes, incluimos un identificador de grupo de imágenes. Así pues, podemos construir una fórmula que responda a la pregunta:

Dada una imagen de referencia RI y las imágenes iniciales AI_1, AI_2, AI_3 y AI_4, correspondientes al paquete P, ¿obtendré siempre una imagen de la vista X, una imagen de la vista Y y una imagen de la vista Z para el paquete de imágenes P (Package01)?

En la pregunta anterior es fácil identificar algunos operadores lógicos ordinarios y otros temporales. Así, podríamos traducir dicha pregunta al lenguaje CTL de lógica temporal en los siguientes términos intuitivos: el operador

CTL $AF(p)$ indica que la propiedad p se verificará eventualmente en el futuro (AF se puede leer como *Always in the Future*, en toda ejecución), a partir del estado inicial. Por lo tanto, estamos preguntando si desde estado inicial del sistema se alcanza un estado del sistema en el que existen las imágenes convertidas *Package01_SliceX_Converted*, *Package01_SliceY_Converted* y *Package01_SliceZ_Converted*, correspondientes al paquete de imágenes *Package01*:

```
AF( AND (RDFG_X, AND(RDFG_Y,RDFG_Z) ) )
RDFGP_X: t:Package01  t:hasConverted  t:Package01_SliceX_Converted
RDFGP_Y: t:Package01  t:hasConverted  t:Package01_SliceY_Converted
RDFGP_Z: t:Package01  t:hasConverted  t:Package01_SliceZ_Converted
```

Tras ejecutar el *Model Checker* para verificar dicha fórmula contra el RG generado anteriormente, obtenemos la siguiente salida:

```
INFO [main] (CombasApp.java:239) - Model satisfies the formula!
INFO [main] (CombasApp.java:244) - Output files generated.
INFO [main] (CombasApp.java:248) - Checked in 13471 millis
INFO [main] (CombasApp.java:249) - Formula: formula_zi0lxko6oc16
INFO [main] (CombasApp.java:250) - Model: netId1lda11bfzilll_RG
```

Lo cual confirma que el modelo satisface la propiedad. Si se aplica al modelo que incluye dos paquetes de imágenes, la propiedad también se verifica. El tiempo requerido para resolver la pregunta anterior se muestra en la Tabla 2.

Caso	**Tiempo de verificación**	**Resultado**
1 paquete de imágenes	13 seg	modelo válido
2 paquetes de imágenes	32 min	modelo válido

Tabla 2. Resultados de la ejecución del *model checker* para el ejemplo

También es posible verificar propiedades de seguridad, en el sentido de asegurar que algo malo no va a ocurrir, mediante la *no verificación* de fórmulas. Para ello, podríamos comprobar si existe algún camino tal que se obtenga alguna imagen convertida final que no contenga información de las imágenes origen. Pasemos por tanto a estudiar la construcción de dicha fórmula. Análogamente al operador AF (en toda ejecución), existe el operador $EF(p)$ (*there Exists an execution*, hay al menos una ejecución) para especificar si existe una ejecución posible que cumpla p en el futuro.

Así podemos construir la fórmula:

```
EF(AND(NOT(RDFGP_atlas),OR(RDFGP_x,OR(RDFGP_y,RDFGP_z))))
```

donde

```
== PROPIEDADES ==
RDFGP_x :
  ?package t:hasConverted ?converted .
  ?converted rdf:type "XviewGIF"
RDFG_y :
  ?package t:hasConverted ?converted .
  ?converted rdf:type "YviewGIF"
RDFG_z:
  ?package t:hasConverted ?converted .
  ?converted rdf:type "ZviewGIF"
RDFGP_atlas:
  ?package t:hasConverted ?converted .
  ?converted t:hasSlice ?slice .
  ?slice t:hasAtlas ?atlas .
  ?atlas t:hasRefImg ?refimg .
  ?atlas t:hasAnatomyImg1 ?img1 .
  ?atlas t:hasAnatomyImg2 ?img2 .
  ?atlas t:hasAnatomyImg3 ?img3 .
  ?atlas t:hasAnatomyImg4 ?img4 .
```

Esta fórmula establece si existe algún camino con un estado futuro que cumpla lo siguiente: *no existe información de origen y, además, se ha obtenido la imagen final de la vista X, Y o Z.* Si el resultado es cierto, nuestro sistema no funcionará de forma correcta. De modo que esperamos que el resultado sea *falso*. Tras ejecutar el *model checker* para verificar dicha fórmula, obtenemos la siguiente salida:

```
ERROR [main] (CombasApp.java:241) - Model doesn't satisfy the formula.
INFO [main] (CombasApp.java:244) - Output files generated.
INFO [main] (CombasApp.java:248) - Checked in 23300 millis
INFO [main] (CombasApp.java:249) - Formula: formula_1pydnao80fxve
INFO [main] (CombasApp.java:250) - Model: netId1lda11bfzilll_RG
```

Lo cual confirma que el modelo no satisface la propiedad, lo que quiere decir que nuestro modelo es válido. El tiempo requerido para verificar esta fórmula en los dos grafos considerados se muestra en la Tabla 3.

Caso	Tiempo de verificación	Resultado
1 paquete de imágenes	23 seg	modelo válido
2 paquetes de imágenes	58 min	modelo válido

Tabla 3. Resultados de la ejecución del *model checker* para el ejemplo con una fórmula alternativa

4. Conclusiones

En este trabajo se ha presentado COMBAS, un entorno completo para el procesamiento, *model checking* de PNS y verificación/visualización de los resultados mediante un entorno visual y adaptado a la utilización de RDF y CTL. La carencia de herramientas de *model checking* que soporten la inclusión de información semántica RDF en el modelo de origen y que no integren las herramientas necesarias para su posterior procesamiento y análisis provocan que COMBAS represente un enfoque novedoso para abordar problemas de PNS, así como de *workflows* científicos, tal y como se ha demostrado mediante su aplicación al *First Provenance Challenge*. A corto plazo se está estudiando la generación de los grafos de alcanzabilidad en entornos de computación *grid*, la extensión del *model checker* para soportar análisis simbólico y la integración de otras tecnologías semánticas en el entorno.

Bibliografía

1. Ibáñez, M., Álvarez, P., Ezpeleta, J.: Predicción simbólica del comportamiento en ejecución de los procesos de negocio semánticos. In: V Jornadas Científico-Técnicas en Servicios Web y SOA – JSWEB'09. (2009) 183–196
2. Kummer, O., Wienberg, F.: Renew - the Reference Net Workshop. In: Tool Demonstrations, 21st International Conference on Application and Theory of Petri Nets, Computer Science Department, Aarhus University, Aarhus, Denmark (2000) 87–89
3. Garlan, D., Khersonsky, S., Kim, J.S.: Model checking publish-subscribe systems. In: Proceedings of the 10th international conference on Model checking software. SPIN'03, Berlin, Heidelberg, Springer-Verlag (2003) 166–180
4. Duflot, M., Fribourg, L., Hérault, T., Lassaigne, R., Magniette, F., Messika, S., S.Peyronnet, Picaronny, C.: Probabilistic model checking of the csma/cd protocol using prism and apmc. Electr. Notes Theor. Comput. Sci. **128** (2005) 195–214
5. Grumberg, O., Veith, H., eds.: 25 Years of Model Checking - History, Achievements, Perspectives. In Grumberg, O., Veith, H., eds.: 25 Years of Model Checking. Volume 5000 of Lecture Notes in Computer Science., Springer (2008)
6. Lomuscio, A., Qu, H., Raimondi, F.: Mcmas: A model checker for the verification of multi-agent systems. In: CAV. (2009) 682–688
7. Moreau, L. et al: Special issue: The first provenance challenge. Concurr. Comput.: Pract. Exper. **20** (2008) 409–418
8. fMRI Data Center: Web del archivo de recursos anatómicos en alta resolución del *fMRI Data Center*. http://www.fmridc.org/ (2010)
9. AIR (*Automated Image Registration*): juego de herramientas AIR de tratamiento de imágenes 3D anatómicas. http://air.bmap.ucla.edu/AIR5/index.html (2010)
10. FSL: conjunto de herramientas FSL de tratamiento de imágenes anatómicas en 3D. http://www.fmrib.ox.ac.uk/fsl/ (2010)
11. Head, D., Snyder, A., Girton, L., Morris, J., Buckner, R.: Frontal-hippocampal double dissociation between normal aging and alzheimer's disease. Cereb Cortex **15** (2005) 732–9

Summary Of "Defining Process Performance Indicators: An Ontological Approach" [1]

A. del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés

Universidad de Sevilla, Spain

Business Process Management (BPM) intends to support business processes using methods, techniques, and software to design, enact, control, and analyse operational processes involving humans, organizations, applications, documents and other sources of information [2]. Many companies are taking this process oriented perspective in their business, as a way of identifying how to improve, where to increase quality, reduce waste or save time in their processes, making thus the evaluation of business processes performance a key aspect.

Performance requirements on business processes can be specified by means of Process Performance Indicators (PPIs) with target values that must be reached in a certain period. A PPI is a measure that reflects the critical success factors of a business process defined within an organisation, in which its target value reflects the objectives pursued by the organization with that business process.

In this paper, we argue the importance of integrating the management of PPIs into the whole business process lifecycle [3, 4] and propose to do it as follows: in the design and analysis phase, PPIs should be modelled together with the business process. Furthermore, this model of PPIs should also enable their analysis by detecting the dependencies amongst them at design time and also using them as part of the business process analysis, for instance in business process simulation techniques. During the configuration phase, the instrumentation of the processes that are necessary to take the measures must be defined. During the business process enactment, when valuable execution data is gathered, the PPIs' values have to be calculated and the monitoring of these PPIs should be carried out. For instance, this can be done based on execution logs that store information about the process such as the start or end of activities. Finally, during the evaluation phase, the monitoring information obtained in the previous phase will help to identify correlations and predict future behaviour.

Handling business processes and the aforementioned PPIs lifecycle is recognized as an error-prone and time-consuming activity which requires of automated support. Current research efforts in the automated treatment of business processes mainly focus on the analysis post-mortem applying technics as process mining. These operations allow us to observe the properties of the process models and extract information about them after their enactment. However, although there are several research proposals to define PPIs, none of them are well-suited because they cannot express commonly used PPIs or they are not ready to enable a design-time analysis of PPIs or they do not define explicitly their relationship with the business process and, hence, make it difficult their use together with business process analysis techniques. Thus, the definition and automated design-time analysis of these PPIs still prevail as open research issues.

To overcome this issue, we present an ontology to define PPIs whose main benefits can be summarised as follows: 1. The relation between PPIs and the business process is explicitly established. This enables the use of PPIs together with other business process analysis techniques and helps in the instrumentation of the information systems that is necessary to obtain measures automatically. 2. It supports the definition of a wide variety of PPIs, including those associated with data objects. It also supports the definition of an expressive analysis period of a PPI. In fact, our ontology supports the definition of PPIs that, as far as we know, cannot be expressed in any other similar proposal. 3. Dependencies between ProcessMeasures and InstanceMeasures can be automatically obtained from the ontology, which enables the analysis of PPIs at design time. Furthermore, since the ontology has been defined in OWL DL, automated reasoners can be used to make queries about the PPI model such as how many PPIs are defined on the same MeasureDefinition? Or how many PPIs are defined on a TimeMeasure?

Furthermore, we have validated the suitability of the ontology for the definition of real PPIs by using several real scenarios in different environments (the Information Technology Department of the Andalusian Health Service and the Justice and Public Administration Department of the Andalusian Local Government), in order to prove the applicability of our solution to actual scenarios. These real scenarios and the corresponding definitions of PPIs using our ontology can be found at `http://www.isa.us.es/ppiontology/`

References

1. del Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Defining Process Performance Indicators: An Ontological Approach. In Proceedings of the 18th International Conference on Cooperative Information Systems (CoopIS), OTM 2010, Part I, LNCS 6426, 555-572 (2010)
2. van der Aalst, W., ter Hofstede, A.,Weske, M.: Business process management: A survey. In: Business Process management. Volume 2678, Springer (2003) 1–12
3. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
4. del Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Towards modelling and tracing key performance indicators in business processes. In: II Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS 2009). (2009)

Quality Assessment of Business Process Models based on Thresholds

Laura Sánchez-González[1], Félix García[1], Jan Mendling[2], Francisco Ruiz[1]

[1] Grupo Alarcos, Universidad de Castilla La Mancha, Paseo de la Universidad, nº4, 13071 Ciudad Real, España
{laura.sanchez | felix.garcia | francisco.ruizg}@uclm.es

[2] Humboldt-Universität zu Berlin, Unter den Linden 6, D-10099 Berlin, Germany, jan.mendling@wiwi.hu-berlin.de

Process improvement is recognized as the main benefit of process modelling initiatives. Quality considerations are important when conducting a process modelling project. While the early stage of business process design might not be the most expensive ones, they tend to have the highest impact on the benefits and costs of the implemented business processes. In this context, quality assurance of the models has become a significant objective. An important step towards improved quality assurance is a precise assessment of quality. We analyze quality from the perspective of understandability and modifiability, which are both sub-characteristics of usability and maintainability, respectively.

Several initiatives about business process metrics were published [2]. The significance of these metrics relies on a thorough empirical validation of their connection with quality attributes. There are, to date, still rather few initiatives to investigate the connection between structural process model metrics and quality characteristics, so we detect a gap in this area which needs more empirical research. Experimental data is obtained by a total of six experiments: three to evaluate understandability and three to evaluate modifiability.

In accordance with the previously identified issues, the purpose of this paper is to contribute to the maturity of measuring business process models. The aim of our empirical research approach is to validate the connections between an extensive set of metrics (number of nodes, diameter, density, coefficient of connectivity, average gateway degree, maximum gateway degree, separability, sequentiality, depth, gateway mismatch, gateway heterogeneity, cyclicity and concurrency) and the ease with which business process models can be understood (understandability) and modified (modifiability). A correlation analysis and a regression estimation were applied in order to test the connection between the metrics and both the understandability and modifiability of the models. After the selection of the most suitable metrics for understandability and modifiability, we extracted threshold values in order to evaluate the measurement results. Such thresholds are an important aid to support the modeller of a business process.

The statistical analyses suggest rejecting the null hypothesis (***H0****: There is no correlation between structural metrics and understandability and modifiability)*, since the structural metrics apparently seem to be closely connected with understandability and modifiability. For understandability these include Number of

Nodes, Gateway Mismatch, Depth, Coefficient of Connectivity and Sequentiality. For modifiability Gateway Mismatch, Density and Sequentiality showed the best results. On the other hand, Table 2 shows the regression equations selected.

Table 1. Prediction models of understandability and modifiability

	Attr.	Experi.	Prediction model	p-value	MMRE	p(0.25)	p(0,30)
Time	U	E3	T3 = 47.04 + 2.46 nºnodes	.000	**.32**	**.51**	**.58**
	M	E4	E4 = 50.08 + 3.77 gateway mismatch + 422.95 density	.000	**.37**	**.31**	**.38**
Correct Answers	U	E2	CA2 = 3.17 - 0.005 nºnodes - 0.38 coeff. of connectivity + 0.17 depth - 0.015 gateway mismatch	.000	**.18**	**.79**	**.79**
	M	E4	CA4 = 1.85 - 3.569 density	.000	**.23**	**.82**	**.83**
Effi-cien-cy	U	E3	EF3 = 0.042 - 0.0005 nºnodes + 0.026 sequentiality	.000	**0.84**	**.22**	**.25**
	M	E4	EF4 = 0.006 + 0.008 sequentiality	.000	**.62**	**.32**	**.42**

After analyzing which measures are most useful, it is interesting to know what values of these measures indicate poor quality in models. That means, thresholds values could be used as an alarm of detecting low-quality structures in conceptual models. Extended results are depicted in [1]. The threshold values could be interpreted as follows: if number of nodes of a model is between 30 and 32, gateway mismatch is between 0 an 2, depth is 1, connectivity coefficient is 0,4 and sequentially is between 0,7 and 0,84 the probability of considering the model efficient in understandability tasks is about 70%, which means model has an acceptable level of quality.

This work has implications both for research and practice. The strength of the correlation of structural metrics with different quality aspects (up to 0.85 for gateway heterogeneity with modifiability) clearly shows the potential of these metrics to accurately capture aspects closely connected with actual usage. Moreover, we demonstrated how threshold values for selected measures can be found, and related these values to different levels of quality related to understandability and modifiability for business process models. From a practical perspective, these structural metrics can provide valuable guidance for the design of process models, in particular for selecting semantically equivalent alternatives that differ structurally. In future research we aim to contribute to the further validation and applicability of process model metrics.

References

[1] Sánchez-González, L., F. García, et al. (2010). "Quality Assessment of Business Process Models Based on Thresholds." CoopIS 2010 - 18th International conference on Cooperative Information Systems: 78-95.

[2] Sánchez-González, L., F. García, et al. (2010). "Measurement in Business Processes: a Systematic Review." Business process Management Journal 16(1): 114-134.

Verification of Real-Time Systems Design *

M. Emilia Cambronero, Valentín Valero and Gregorio Díaz
emicp@dsi.uclm.es, valentin@dsi.uclm.es, gregorio@dsi.uclm.es

Dept. Sistemas Informáticos
Universidad de Castilla-La Mancha, 02071 Albacete, Spain

Abstract. The main objective of this paper is to present an approach to accomplish the verification in the early design phases of a system, which allows us to make the system verification easier, specifically for those systems with timing restrictions. For that purpose we use RT-UML sequence diagrams in the design phase and we translate these diagrams into timed automata for performing the verification by using model checking techniques.

1 Introduction

We present in this paper a part of the methodology shown in [1], specifically the translation from RT-UML sequence diagrams into timed automata, with the purpose of system design verification.

For that purpose we use UML 2.0 [1], which provides us some great mechanisms that when combined with Model-Driven Development capabilities, enable the building of complete and fully tested reusable components. These mechanisms are frames, which can be used jointly with the UML profile for Schedulability, Performance and Time [1] (also called RT-UML) in order to describe control structures and timing restrictions in our systems. This profile was introduced to cover the lacks of UML in some key areas that are of particular concern to real-time system designers and developers. Specifically, we use RT-UML sequence diagrams, which are one of the central diagram types used in RT-UML. These allow us to describe the behavior of a group of entities over time, thus facilitating the understanding of their collaborative behavior, by capturing their interactions over time. Likewise, by using timed automata, we have a well-known mathematical formalism that allows us not only to describe but also to analyze the behavior of discrete dynamical systems with timing restrictions. In this paper, then, we present a method which avoids this learning cost, modeling the dynamic behavior of a timed system by RT-UML, enabling this description to be then translated automatically to timed automata and verified by using the model-checker of the UPPAAL tool [1].

2 Translating RT-UML documents into Timed Automata

Since the syntax of RT-UML is so vast, as described in [1], we need to restrict ourselves to a subset of it in order to define a translation to NTAs. Thus, only

* Supported by the Spanish government (co-financed by FEDER founds) with the project TIN2006-15578-C02-02, and the JCCLM regional project PAC06-0008-6995.

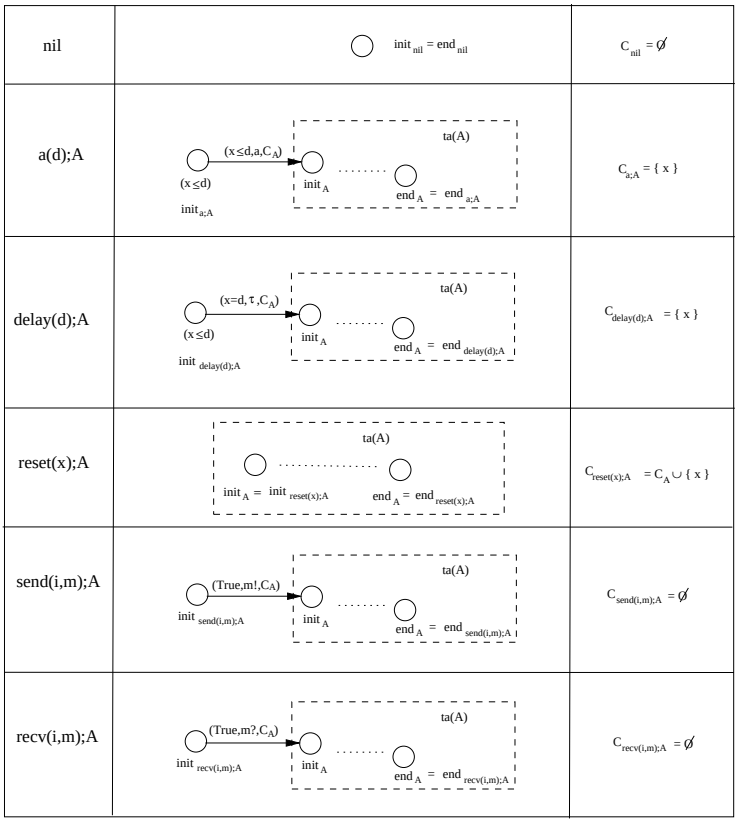


Fig. 1. Activity translation

the elements that have been considered in the meta-language presented in [1] are considered in the translation.

We first define the timed automaton associated to each activity, and afterwards, the NTA associated to every sort of frame. The translation for the different activities is shown in Figure 1.

The translation for the different frames is shown in [1].

2.1 Conclusions

This paper shows how a real-time system description written by using RT-UML sequence diagrams can be automatically translated into a network of timed automata. For that purpose we have defined a meta-model for RT-UML sequence diagrams, and an operational semantics for this meta-model. We have also defined the formal semantics of timed automata and a translation from the meta-model to NTA has been defined, which has been proved to be correct, in the sense that both semantics yield the same timed computations.

References

1. M. E. Cambronero, V. Valero and G. Diaz. Verification of Real-Time Systems Design. *Sofware Testing Verification and Reliability*, 20:3–37, 2010.

Sesión 3: Arquitecturas SOA

Procesamiento de Eventos Complejos en Entornos SOA: Caso de Estudio para la Detección Temprana de Epidemias

Juan Boubeta Puig, Guadalupe Ortiz Bellot e Inmaculada Medina Bulo

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Cádiz,
C/Chile 1, 11002 Cádiz, España
{juan.boubeta,guadalupe.ortiz,inmaculada.medina}@uca.es

Resumen Las arquitecturas SOA han emergido como una solución eficiente para la implantación de sistemas en los que la modularidad y las comunicaciones entre terceros son un factor clave; sin embargo, no son adecuadas para aquellos sistemas en los que hay necesidad de detectar situaciones relevantes o excepcionales en tiempo real. En este sentido, el procesamiento de eventos complejos (CEP) es una tecnología emergente que permite procesar y correlacionar continuamente grandes cantidades de eventos para detectar y responder a las condiciones cambiantes de los procesos de negocio. En este artículo proponemos una arquitectura que integra CEP con SOA y se aplica a un caso de estudio para la detección temprana de brotes epidemiológicos de gripe aviar. Los resultados confirman que CEP es idóneo para decrementar considerablemente el tiempo que transcurre desde que existe un caso de epidemia hasta que se genera una alarma de aviso, reduciéndose así el impacto de enfermedades.

Keywords: CEP, patrones de eventos complejos, SOA 2.0, ESB, salud pública.

1. Introducción

En los últimos años, las arquitecturas orientadas a servicios o *Service-Oriented Architecture* (SOA) han surgido como una solución eficiente para la implantación de sistemas en los que la modularidad y las comunicaciones entre terceros son un factor clave, desarrollándose así aplicaciones distribuidas compuestas de componentes (servicios) reutilizables y compartibles. Estos componentes disponen de unas interfaces bien definidas independientes de sus implementaciones, lo que permite que estos sistemas puedan adaptarse fácil y rápidamente a las condiciones cambiantes del negocio. Sin embargo, estas arquitecturas no son adecuadas para aquellos entornos en los que hay necesidad de analizar continuamente toda la información que fluye por el sistema para detectar cuanto antes y de forma automática las situaciones que son críticas para los procesos de negocio.

Esta limitación puede suplirse con el uso añadido del procesamiento de eventos complejos o *Complex Event Processing* (CEP) [15], una tecnología que proporciona un conjunto de técnicas que ayudan a hacer un uso eficiente de las

arquitecturas dirigidas por eventos o *Event-Driven Architecture* (EDA). CEP permite procesar y analizar grandes cantidades de eventos, así como correlacionarlos para detectar y responder en tiempo real a situaciones críticas del negocio. Para ello, se utilizan unos patrones de eventos que inferirán nuevos eventos más complejos y con un mayor significado semántico, que ayudarán a tomar decisiones ante las situaciones acontecidas.

Así pues, combinando el uso de SOA con CEP, podremos detectar eventos relevantes en sistemas complejos y heterogéneos. En la actualidad, la integración de EDA y SOA se conoce como SOA dirigida por eventos (ED-SOA) o SOA 2.0 [18], una extensión de SOA para responder a los eventos que ocurren como resultado de los procesos de negocio. Este nuevo enfoque hará posible que los servicios no intercambien únicamente mensajes entre ellos, sino que también puedan publicar eventos y recibir notificaciones de eventos de distintos servicios. Para lograrlo, se requerirá un bus de servicios empresariales o *Enterprise Service Bus* (ESB) que actuará como capa de integración para la transformación, el enriquecimiento y el encaminamiento de mensajes entre servicios de diferentes aplicaciones.

Hasta el momento, no se han propuesto arquitecturas que integren adecuadamente SOA, EDA, CEP y la detección de patrones complejos para solucionar este problema: existen propuestas que usan aproximaciones no estándares para la integración de SOA y EDA [14,20] o incluso otras que emplean motores de reglas [19]. Estas aproximaciones que usan motores de reglas son más lentas y menos eficientes en el manejo y recepción de notificaciones, frente a aquellas que emplean motores CEP [12]. Además no tienen en cuenta la exhaustiva cantidad de eventos que el sistema puede llegar a manipular en un momento dado, provocando un fuerte impacto en el rendimiento del sistema.

En este artículo se propone una arquitectura software que integra SOA 2.0 y CEP para la detección de eventos complejos. Esta arquitectura utilizará un ESB beneficiándose de todas sus ventajas y un motor de eventos que facilitará la detección de situaciones relevantes de forma eficiente. Una de las características principales de esta arquitectura es que el bus se encargará de priorizar los eventos según su tipo; de esta forma, evitaremos que se produzca el efecto cuello de botella en el motor CEP, que analizará en primer lugar, de entre todos los eventos recibidos en un instante de tiempo concreto, aquellos que tengan mayor prioridad. Además, esta arquitectura se diferencia de otras propuestas en la utilización de bases de datos NoSQL (*Not only SQL*) [8], un tipo de base de datos emergente basada en relaciones *clave/valor*, fácilmente escalables horizontalmente y eficientes para la manipulación de grandes cantidades de datos.

Finalmente, para una mejor comprensión de la arquitectura propuesta este artículo describe la implementación realizada de un caso de estudio para la detección temprana de brotes epidemiológicos que ilustra su utilidad.

El resto del artículo se estructura de la siguiente forma. En la Sección 2 se describen las principales características de CEP y se compara con SOA. A continuación, en la Sección 3 se propone una arquitectura software que integra SOA 2.0 con CEP. En la Sección 4 se describe e implementa un caso de estudio en

el que se emplea esta arquitectura para la detección en tiempo real de epidemias y pandemia de gripe aviar. Además, en la Sección 5 se presentan los resultados obtenidos del estudio. Por otro lado, en la Sección 6 se resumen algunos trabajos en los que se integran CEP con SOA. En la Sección 7 se detallan las ventajas que presenta la arquitectura propuesta sobre otras existentes. Por último, se presentan las conclusiones y el trabajo futuro.

2. Procesamiento de Eventos Complejos

CEP [15] es una tecnología que proporciona un conjunto de técnicas que ayudan a hacer un uso eficiente de las arquitecturas EDA. Se basa en el filtrado de eventos irrelevantes y en el reconocimiento de patrones de los eventos que sí son relevantes. Además, hace posible la detección de nuevos eventos más complejos, conocidos como "situaciones"; es decir, CEP permite la captura y correlación de cualquier tipo de evento con el fin de detectar situaciones de una mayor complejidad semántica e inferir conocimiento valioso para los usuarios finales.

La característica principal de estos eventos complejos procesados mediante tecnología CEP es que pueden ser identificados e informados en tiempo real, reduciendo la latencia en la toma de decisiones; a diferencia del software tradicional de análisis de eventos, que no funciona en tiempo real.

Así pues, CEP es una tecnología fundamental para aplicaciones que deban responder a situaciones que cambien rápidamente y de forma asíncrona —donde las interacciones no tienen que ser transacciones—, gestionar excepciones o reaccionar rápidamente a situaciones inusuales; y que requieran adaptabilidad y bajo acoplamiento.

CEP tiene algunas similitudes y diferencias con SOA. En cuanto a las similitudes, ambos enfoques presentan modularidad, bajo acoplamiento y flexibilidad. Algunas de las diferencias principales son las siguientes. Por un lado, en SOA la interacción está basada en servicios —un usuario debe conocer cuál es el productor del servicio y la interfaz para enviarle las peticiones—, mientras que CEP aplicado a EDA es reactivo y más desacoplado, ya que los eventos son generados por los productores de eventos y serán los consumidores los que se encarguen de interceptarlos y procesarlos. Por otro lado, mientras que los procesos SOA utilizan eventos para dirigir el flujo de control [13] —estos procesos pueden tanto enviar como recibir eventos—, los motores CEP analizan y correlacionan continuamente estos eventos para evaluar si se cumplen las condiciones definidas en alguno de los patrones de eventos almacenados en estos motores.

3. Arquitectura para la Integración de CEP y SOA 2.0

A continuación se propone una arquitectura que integra CEP y SOA 2.0, esto es, una arquitectura orientada a servicios y dirigida por eventos.

El elemento principal de esta arquitectura es el ESB, que permitirá combinar ambos enfoques. Como puede verse en la Figura 1, los productores de eventos son servicios web, aplicaciones y sensores. Algunas de estas aplicaciones pueden

ser: aplicaciones web que permiten interactuar con los usuarios para la gestión de información del sistema (insertar, eliminar, actualizar y modificar datos) y aplicaciones *legacy*, que aunque son antiguas se necesita seguir utilizando porque su reemplazo supone un gran coste para la empresa. Los sensores son los dispositivos que monitorizan el entorno para captar información (temperatura, luminosidad, lluvia, etc.) que posteriormente es transmitida al sistema mediante el controlador que se encuentra integrado en estos sensores.

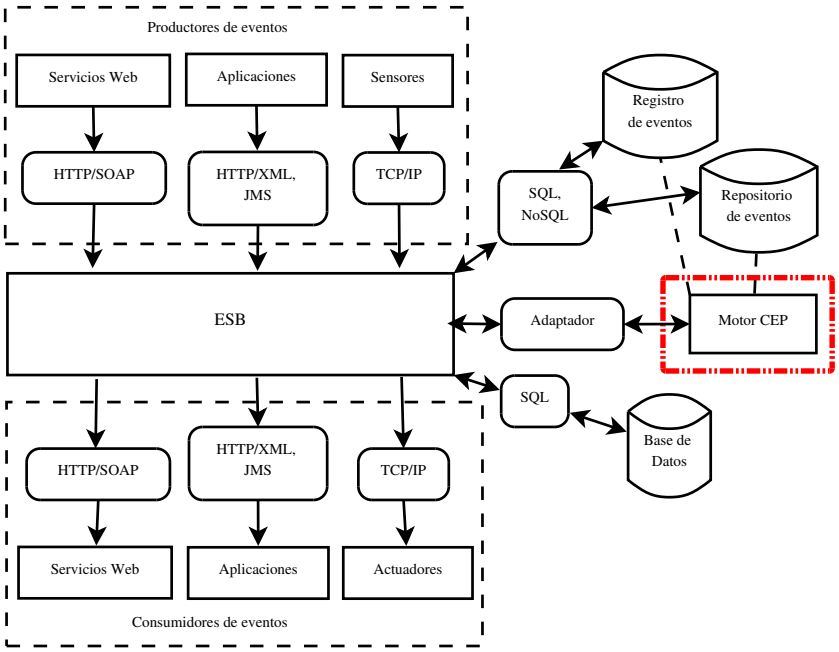


Figura 1. Arquitectura propuesta para la integración de SOA 2.0 y CEP

Para cada tipo de estos productores se utilizará un adaptador que se encargará de transformar cada uno de los mensajes al protocolo que entienda el ESB: HTTP/SOAP para servicios web, HTTP/XML o JMS para aplicaciones y TCP/IP para los sensores.

Seguidamente, estos mensajes modificados se enviarán al ESB. Las tres funcionalidades principales de este bus son:

1. **Transformación de mensajes de eventos**: se transformará el formato nativo de estos eventos a un formato que cumpla un modelo, por ejemplo, un documento XSD (*XML Schema*), que se encontrará almacenado en el *registro de eventos*, una base de datos que contendrá información relevante sobre los metadatos de los eventos. De esta forma, todos los eventos procedentes de distintos productores serán homogeneizados para facilitar al motor CEP su procesamiento.

2. **Enriquecimiento de mensajes de eventos**: una vez transformados, se priorizarán los eventos según el orden que se encuentre establecido para cada tipo de evento en el *registro de eventos*. Esta priorización de eventos evitará que se produzca el efecto cuello de botella en el motor CEP. Así pues, si existe una cantidad exhaustiva de eventos para un instante de tiempo concreto, el sistema atenderá primero a aquellos que tengan más prioridad.
3. **Encaminamiento de mensajes de eventos**: el bus se encargará de enviar los eventos a todas las partes que estén interesadas en recibirlos y previamente se hayan suscrito.

Una vez transformados y enriquecidos, los eventos se almacenarán en el *repositorio de eventos*, una base de datos que se utilizará como el histórico de eventos. Este repositorio permitirá el procesamiento retrospectivo al motor CEP, esto es, realizar consultas sobre eventos pasados. Tanto el repositorio como el registro de eventos serán base de datos NoSQL, debido a que son propicias para la gestión de grandes cantidades de datos, entre otras, como se comenta en la Sección 7. Sin embargo, si el motor CEP no permite la interacción con bases de datos NoSQL, se utilizarán bases de datos SQL. Además, esta arquitectura podrá implantarse en organizaciones que dispongan de sistemas *legacy*, para ello se contempla la posibilidad de conectar también bases de datos tradicionales SQL al bus.

A la vez que los eventos se envían al sistema de gestión de base de datos para almacenarlos, también serán enviados e introducidos en los flujos de eventos del motor CEP. Este motor contendrá los patrones de eventos que especificarán las condiciones que deben cumplirse para detectar las situaciones que son relevantes y las acciones que se llevarán a cabo. Algunas de las funciones de este motor son: filtrar (eliminar del flujo de eventos aquellos que no son de interés), correlacionar y fusionar eventos de distintos flujos (podrían crearse eventos complejos), y agregar (agrupar los eventos de un flujo). Las alarmas (eventos complejos) que se generen serán publicadas inmediatamente en el ESB.

Finalmente, se notificarán estos eventos a aquellos consumidores de eventos que se hayan suscrito. Estos consumidores serán servicios web, aplicaciones —como cuadros de mando que permitirán visualizar las alarmas—, o actuadores que realizarán alguna acción (apagado/encendido, apertura/cierre, etc.) sobre un dispositivo determinado.

4. Caso de Estudio

Debido a que en las últimas décadas la globalización ha llevado consigo un gran aumento de desplazamientos de personas entre distintos países, ha aumentado drásticamente el impacto de las epidemias de enfermedades emergentes, convirtiéndose en la actualidad en una amenaza importante para la vida, la seguridad y la economía mundial [17]. Por este motivo, hemos implementado un caso de estudio para detectar epidemias de gripe aviar en tiempo real.

En este caso de estudio se ha adaptado la arquitectura que integra CEP y SOA 2.0, introducida en la Sección 3, para detectar brotes epidemiológicos de gripe aviar. En concreto, este sistema de salud pública permitirá lanzar alertas

en tiempo real cuando se detecten casos sospechos de pacientes que pueden estar infectados por este virus, casos confirmados de estos pacientes sospechosos, casos epidémicos —países que sufren la epidemia de gripe aviar— y el caso pandémico a nivel mundial —la epidemia afecta a varios países— (véase la documentación de la *Organización Mundial de la Salud* (OMS) [9] para obtener más información sobre la definición de estos casos).

Los productores de eventos serán los siguientes:

- **Hospitales**: el personal sanitario, especialmente los médicos, diagnosticará síntomas y obtendrá información relevante sobre los pacientes, y emitirán eventos mediante la inclusión de los diagnósticos de los pacientes en los sistemas de información del hospital.
- **Laboratorios**: a partir de análisis de sangre y otras técnicas, los laboratorios podrán detectar casos confirmados de gripe aviar y deberán publicar esta información dentro de sus sistemas de información, lo que dará lugar a los eventos correspondientes.

Por otro lado, los consumidores de eventos serán:

- **La OMS y otras organizaciones internacionales**: estas organizaciones se suscribirán para recibir eventos relevantes sobre brotes epidemiológicos y estar al corriente de los casos sospechosos, confirmados, epidémicos y pandémicos que se hayan detectado a nivel mundial.
- **Hospitales**: el personal sanitario, especialmente los médicos, también necesitarán conocer en todo momento cuáles son los casos que se han detectado para adoptar medidas que permitan paliar esta situación, como puede ser el aislamiento de un paciente.
- **Laboratorios**: los laboratorios tendrán que estar informados en todo momento de la evolución del virus y de su propagación, de cara a la investigación de nuevos antídotos o fármacos que ayuden a combatir la enfermedad.

Tanto los productores como los consumidores de eventos serán servicios, que podrán emitir y/o recibir eventos, así como distintas aplicaciones, por ejemplo cuadros de mando, que permitan a los usuarios finales visualizar gráficamente dichas situaciones. A parte de estos, los emisores también podrán ser sensores que envíen, por ejemplo, eventos con la temperatura actual del ambiente; otros consumidores de eventos podrán ser actuadores, que llevarán a cabo alguna acción sobre un dispositivo concreto cuando se detecte una determinada situación.

Por ejemplo, los servicios de un hospital podrán lanzar un evento complejo si se detecta que un paciente es sospechoso de estar infectado por el virus de la gripe aviar. Este evento será recibido por los servicios de las farmacias y la OMS, que reaccionarán inmediatamente ante esta situación: las farmacias comunicarán automáticamente a sus proveedores un aumento del pedido de aquellos medicamentos que ayuden a combatir dicha enfermedad, y la OMS lanzará alarmas de aviso a aquellos laboratorios y organismos internacionales de salud que estén interesados en conocer esta situación que acaba de acontecer.

El objetivo de este caso de estudio es comprobar si CEP es una solución efectiva para detectar epidemias y pandemias en tiempo real, frente a la mayoría

de las herramientas actuales que alertan de epidemias y pandemias con una periodicidad semanal: Flunet [4] presenta información de la *influenza* de todos los países del mundo y Euroflu [3] solo de los estados miembros de la región europea de la OMS. Esto permitirá que los responsables sanitarios puedan paliar cuanto antes el impacto de epidemias y pandemias a nivel mundial.

Por ello, únicamente nos centraremos en el uso del motor CEP de la arquitectura propuesta, la definición de un conjunto de patrones que se adecúe a nuestro objetivo y la implementación de un simulador de eventos que estará conectado directamente al motor CEP. Este simulador producirá eventos de estados de pacientes aleatorios ante la dificultad de acceder a los sistemas de información oficiales de hospitales, que sustituirá a los productores de eventos de dicha arquitectura. Por tanto, nuestro sistema no dispondrá del ESB; cuya integración forma parte de nuestro trabajo futuro.

Seguidamente se definen los patrones de eventos complejos que permitirán detectar dichas situaciones y se especifican los pasos a seguir para implementar y validar nuestro caso de estudio.

4.1. Patrones de Eventos Complejos Aplicados a la Salud

A continuación, describimos la definición de los patrones de eventos complejos que permiten detectar casos sospechosos, confirmados, epidémicos y pandémicos de gripe aviar. Para ello, se ha adaptado el esquema de los patrones de diseño de Buschmann [11]: el *nombre* del patrón y un resumen, un *ejemplo* real que demuestre la existencia del problema y la necesidad de aplicar el patrón, las *situaciones* (contexto) en las que el patrón debería ser aplicado, el *problema* que trata el patrón, la *solución* que propone, una especificación detallada de los *aspectos estructurales*, la *implementación* del patrón en un lenguaje específico, así como las *consecuencias* (beneficios y desventajas) que ofrece el patrón.

En este trabajo solo detallaremos el nombre del patrón y su implementación. Estos patrones se han implementado en el lenguaje de procesamiento de eventos complejos de Esper [2], EPL (*Event Processing Language*), por varios motivos. En primer lugar, la curva de aprendizaje no es elevada, ya que su sintaxis se aproxima bastante a la del lenguaje SQL ampliamente conocido a nivel mundial. Por otro lado, EPL soporta de forma nativa varios tipos de formato de eventos: objetos Java/.NET, *maps* y documentos XML. Además, permite la personalización del lenguaje así como del motor Esper, escrito en Java y de código abierto.

Caso Sospechoso de Gripe Aviar El patrón *SospechosoGripeA* permite detectar posibles apariciones de casos de gripe aviar, cuando se cumplan las siguientes condiciones:

1. El paciente tenga fiebre (más de 38 ^{o}C) o tos o cefalea o mialgias o conjuntivitis o faringitis o encefalopatía o fracaso multiorgánico o neumonía.
2. Y, además, presente una historia de exposición a fuente de contagio conocida en período de contagio (7 días previos):
   a) Estancia en zona donde se han reportado casos humanos de gripe aviar.

b) Haber tenido contacto con una persona ya diagnosticada de gripe aviar.
c) Haber tenido contacto con animales que pueden estar infectados.
d) Haber trabajado en un laboratorio manipulando gases.

La implementación del caso sospechoso de gripe aviar en EPL es la siguiente:

```
insert into SospechososGripeA
select sospechosoGripeA.id, sospechosoGripeA.tiempoRegistro,
  sospechosoGripeA.paciente.sexo, sospechosoGripeA.ubicacionActual
from pattern [every sospechosoGripeA = EstadoPaciente((tos or
  temperatura > 38 or cefalea or mialgias or fracasoMultiorganico
  or faringitis or conjuntivitis or encefalopatia or neumonía)
  and ((EstadoPaciente.NumDias(tRegistro, estanciaZonaInfectada)
    <= 7) or (EstadoPaciente.NumDias(tRegistro,
    contactoPersonaContagiada) <= 7) or (EstadoPaciente.NumDias(
    tRegistro, contactoAnimalSospechoso) <= 7) or (EstadoPaciente.
    NumDias(tRegistro, exposicionGasesLaboratorio) <= 7))]
```

En primer lugar, se define el patrón de eventos complejos mediante la cláusula `from pattern`. Se toma de entre todos los eventos `EstadoPaciente` del flujo de datos aquellos que cumplan las condiciones de caso sospechoso descritas anteriormente, y se le aplica el operador `every` para seleccionar cada uno de estos pacientes que sean sospechosos de gripe aviar (a los que se les asignan el alias `sospechosoGripeA`). `NumDias` es una función definida para calcular el número de días existentes entre la fecha en la que se ha registrado el evento de tipo `EstadoPaciente` y la fecha en la que el paciente estuvo en contacto con una fuente de riesgo, si se produjo ese contacto.

A continuación, se seleccionan el identificador, el tiempo de registro, el sexo y la ubicación actual de cada uno de estos eventos complejos `sospechosoGripeA` y se insertan, mediante la cláusula `insert into`, en un nuevo flujo de eventos denominado `SospechososGripeA`.

Para procesar las notificaciones de los eventos asociados a este patrón se utilizará un *listener* específico (`SospechosoGripeAListener`), que avisará de que se ha detectado un caso sospecho de gripe aviar.

Caso Confirmado Existe un caso sospechoso de gripe aviar y el laboratorio confirma a lo largo de la semana que el paciente está infectado.

Caso de Epidemia Existen 25 o más casos confirmados de gripe aviar en un país concreto durante una semana.

Caso de Pandemia Existen 2 o más casos de epidemia durante una semana.

4.2. Implementación y Prueba

Los pasos que se han seguido para implementar y probar el caso de estudio, utilizando el lenguaje Java y el motor Esper, son:

1. Definición de las clases Java que representan a los dos tipos de eventos del caso de estudio: *EstadoPaciente* y *VirusGripeAviar*. Los atributos de *EstadoPaciente* son: un identificador único, la fecha, la hora y el lugar en el que ha sido atendido el paciente, el sexo, los síntomas que presenta, así como la fecha en la que haya estado expuesto a algún riesgo. Los atributos de *VirusGripeAviar* son: la fecha, la hora y el laboratorio en el que se ha detectado el virus junto con el identificador del paciente infectado.
2. Creación de un generador de eventos *EstadoPaciente* para simular pacientes atendidos en todo el mundo y el estado en el que se encuentra cada uno de ellos; ante la limitación de disponer de estados de pacientes reales.
3. Configuración e inicialización del motor Esper.
4. Definición y registro de los patrones de eventos en Esper, que se encargarán de detectar los eventos complejos.
5. Implementación de los *listeners* que se encargarán de recibir las notificaciones sobre casos sospechosos, confirmados, epidémicos y pandémicos, y mostrar dicha información a los usuarios interesados. Para ello, será necesario asociar cada *listener* con uno de los patrones de eventos registrados previamente.
6. Introducción de los eventos de tipo *EstadoPaciente* en un flujo de eventos de Esper y los eventos *VirusGripeAviar* en otro flujo distinto.
7. Definición e implementación de los casos de prueba mediante el marco de pruebas unitarias JUnit, y validación de la aplicación.

5. Resultados del Caso de Estudio

En este estudio evaluamos si utilizar un motor CEP en nuestra arquitectura es adecuado, basándonos principalmente en su capacidad para procesar una tasa elevada de eventos.

Se han utilizado eventos simulados y eventos reales: hemos generado 100.000 eventos aleatorios de tipo *EstadoPaciente* y hemos utilizado los 448 casos reales de pacientes infectados de gripe aviar registrados a lo largo del tiempo por distintos laboratorios del mundo; estos datos reales han sido proporcionados por la base de datos EpiFlu™ [5]. Para poder realizar un estudio más detallado, los eventos de cada tipo se han almacenado en un fichero CSV (*Comma-Separated Values*); el motor Esper permite leer múltiples ficheros CSV para simular eventos provenientes de distintos flujos de eventos, así como especificar el número de eventos que se desea enviar por segundo (máximo 1.000 eventos) a cada flujo.

Se han realizado diez ejecuciones enviando desde 100 eventos *EstadoPaciente* y 1 evento *VirusGripeAviar* (el 1 % del número de eventos *EstadoPaciente*), en la primera ejecución, hasta 1.000 eventos *EstadoPaciente* y 10 eventos *VirusGripeAviar*, en la última ejecución. En cada ejecución se han medido el tiempo total (suma de tiempos de CPU, entrada/salida y retardos) necesario para procesar

el número de eventos enviado al motor Esper, así como el tiempo consumido únicamente por la CPU.

Este estudio se ha llevado a cabo en una máquina con un procesador Intel Core™ i7 - 740QM (1,73 GHz) y 4.096 MB de memoria RAM. Los tiempos han sido medidos mediante la aplicación Java VisualVM [6].

En cuanto a los tiempos totales consumidos por el sistema, en la Figura 2 se puede ver que al enviar 100 eventos *EstadoPaciente* y 1 evento *VirusGripeAviar*, por segundo, se han consumido 997.774 ms; mientras que al enviar 1.000 eventos *EstadoPaciente* y 10 eventos *VirusGripeAviar* solo se han consumido 96.879 ms (véase la Figura 3). Esta situación parece bastante lógica: si disponemos de un total de 100.000 eventos *EstadoPaciente* y enviamos 100 eventos/s evidentemente el sistema tardará más que si enviamos 1.000 eventos/s.

Por otro lado, como podemos observar en la Figura 2, para procesar los 100 eventos *EstadoPaciente* y 1 evento *VirusGripeAviar* enviados cada segundo han sido necesarios 43.605 ms de CPU frente a los 83.831 ms consumidos al enviar cada segundo 1.000 *EstadoPaciente* y 100 *VirusGripeAviar* (véase la Figura 3). Aunque estos eventos *EstadoPaciente* han sido generados aleatoriamente a partir de los 100.000 eventos disponibles, estos resultados demuestran que el motor Esper tendrá mayor carga de trabajo cuanto mayor sea la tasa de llegada de eventos; sin embargo a pesar de haber aumentado considerablemente el número de eventos enviado por segundo, podemos ver que el aumento en el tiempo de procesamiento de la CPU no es significativo. Por tanto, podemos afirmar que el motor Esper es una alternativa eficiente para el procesamiento de grandes cantidades de eventos.

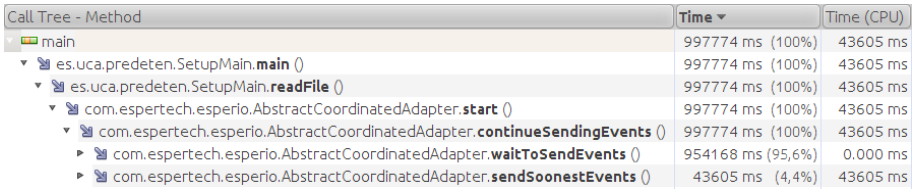


Figura 2. Tiempo total y de CPU consumidos al enviar por cada segundo 100 eventos *EstadoPaciente* y 1 evento *VirusGripeAviar*

6. Trabajos Relacionados

En los últimos años se está considerando CEP como una tecnología fundamental para la detección en tiempo real de situaciones relevantes en distintos contextos: fraudes, compra y venta de acciones, logística y salud, entre otros. Por ejemplo, Velandia-Briñez et al. [21] proponen un método de clasificación basado en CEP para imágenes de mamografías, ayudando a los radiólogos en la detección y prevención temprana del cáncer.

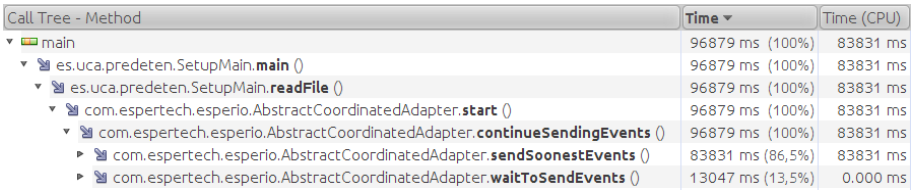


Figura 3. Tiempo total y de CPU consumidos al enviar por cada segundo 1.000 eventos *EstadoPaciente* y 10 eventos *VirusGripeAviar*

En la literatura existen diversos trabajos sobre la integración de CEP y SOA también aplicada a distintos dominios. Ayllón y Reina [10] definen los elementos que pueden aparecer en una arquitectura que permita la integración de CEP en SOA: un *gestor de eventos complejos* para registrar y procesar en tiempo real todos los eventos producidos en el ESB y en base a patrones ser capaz de inferir eventos complejos y generar respuestas automáticas —elemento clave para que pueda integrarse CEP en SOA—, *sistemas legacy*, *servicios web* que proporcionan la funcionalidad de negocio al ESB, un *motor WS-BPEL* para la orquestación de servicios web, un *flujo de trabajo BPM* para el modelado de negocio de aquellos procesos que requieran de intervención humana y *herramientas para analizar y monitorizar en tiempo real* la información que es de interés para una empresa.

Meng et al. [14] implementan un sistema dirigido por eventos basado en identificación por radiofrecuencia para monitorizar los gases emitidos por los vehículos y detectar si la emisión de un vehículo no es estándar, alertando de esta situación a los interesados en proteger el medio ambiente y la calidad del aire. Los autores afirman que en el motor CEP todos los eventos se representan mediante POJO (*Plain Old Java Object*) y que el lenguaje que se utiliza para procesar continuamente los eventos es similar a SQL, sin especificar si se ha extendido para manipular ventanas temporales. Por un lado, en esta propuesta los eventos no se representan en XML, lo que permitiría hacerlos más legibles y reutilizables, como permite el motor Esper [2]. Por otro lado, a diferencia de Esper, tampoco especifican que el motor y el lenguaje puedan ser personalizados.

Taher et al. [20] proponen la adaptación de interacciones de mensajes entre servicios web con interfaces incompatibles. Para ello desarrollan una arquitectura que integra un motor CEP y dispone de adaptadores de entrada y salida de mensajes SOAP. Los adaptadores de entrada recibirán los mensajes enviados por los servicios web, los transformarán a la representación adecuada para que sea manipulado por el motor CEP y se los enviará al motor. Por otro lado, los adaptadores de salida recibirán los eventos producidos por el motor, los transformarán a mensajes SOAP y, entonces, se enviarán a los servicios web. Esta arquitectura presenta algunas limitaciones con respecto a nuestra propuesta. En primer lugar, los servicios interactúan directamente con un marco de trabajo que integra tanto los adaptadores de mensajes como el motor CEP, en lugar de utilizar un ESB que se conecte, por un lado, con los servicios y, por otro lado,

con el motor CEP —el uso de este tipo de bus proporcionará una arquitectura más desacoplada, flexible y reutilizable—. En segundo lugar, estos adaptadores no presentan otras funcionalidades como el encaminamiento de mensajes basado en contenido y la transformación de protocolos que sí presentan los ESB.

Sottara y Colombini [19] proponen una arquitectura para desarrollar una estrategia de control simple para una planta específica de tratamiento de pérdida de agua, en la que el ESB se utiliza para conectar servicios distribuidos en nodos diferentes de forma que sea transparente al usuario. Utilizan la solución JBoss ESB [7] y su integración con el motor de reglas Drools [1], que se encuentra embebido en este bus. Nuestra arquitectura mejora esta al utilizar un motor CEP en lugar de un motor de reglas, como se explica en la Sección 7.

7. Discusión

La arquitectura que se ha propuesto para la integración de SOA 2.0 y CEP se distingue de otras propuestas en la incorporación conjunta de varias mejoras.

En primer lugar, en esta arquitectura se utiliza un motor CEP en lugar de un motor de reglas. Según Chandy y Schulte [12] existen algunas diferencias entre los motores CEP y de reglas, que se detallan a continuación.

Normalmente los motores de reglas son dirigidos por peticiones, es decir, cuando una aplicación necesite tomar una decisión invocará a este motor que llegará a una conclusión a partir de un conjunto de premisas. El modelo general de un motor de reglas es "*Si <alguna condición> entonces <realizar la acción X>*". En la mayoría de las aplicaciones será necesario analizar una gran cantidad de reglas antes de tomar una decisión, convirtiéndose en un problema para tomar decisiones en tiempo real.

Sin embargo, los motores CEP son dirigidos por eventos y se ejecutan continuamente procesando notificaciones de mensajes tan pronto como llegan, de acuerdo con los principios de EDA. La noción de inferencia automática e implícita no existe en la mayoría de estos motores: generan eventos complejos que, a su vez, pueden servir como base para generar otros eventos más complejos, con un mayor contenido semántico. En este caso, en lugar de un modelo *si-entonces-si_no* es una regla —denominada *patrón de eventos complejos—cuando-entonces*: *"cuando <ocurre algo o se detecta alguna condición> entonces <realizar la acción X>*". En un motor CEP, la equivalencia de la cláusula *"si_no*" es la que especifica *"cuando <algo no ha ocurrido en un tiempo especificado> entonces <realizar la acción Y>*". Por tanto, los patrones de eventos utilizan el tiempo como una dimensión más.

Otras ventajas que ofrecen los motores CEP son mayor rapidez y eficiencia en el manejo y recepción de notificaciones, ya que estos gestionan directamente la entrada y salida con los sistemas de mensajería, mientras que los motores de reglas funcionan como servicios que utilizarán las aplicaciones que llevan a cabo la entrada y salida.

En segundo lugar, otra mejora introducida en la arquitectura es la priorización de los eventos según el orden que se haya establecido previamente para cada

tipo de evento. Esta priorización de eventos evitará que se produzca el efecto cuello de botella en el motor CEP, ya que este atenderá primero a aquellos eventos que tengan más prioridad, evitándose así la gestión de una gran cantidad de eventos en un instante de tiempo concreto.

Finalmente, esta arquitectura utiliza bases de datos NoSQL. Las principales diferencias con el tipo de base de datos SQL, que suscita su propuesta como candidata para el almacenamiento de eventos, son las siguientes [16]. Las actualizaciones son asíncronas BASE (*Basically Available, Soft state, Eventual consistency*) en vez de síncronas ACID (*Atomicity, Consistency, Isolation, Durability*). Por otro lado, las bases de datos NoSQL están optimizadas para reaccionar a los cambios —consistencia eventual—, no para gestionar transacciones, y no requiren la definición de sus esquemas ni los tipos de datos. Además son distribuidas, fácilmente escalables horizontalmente y muy eficientes para la manipulación de grandes cantidades de datos.

8. Conclusiones y Trabajo Futuro

En este trabajo se ha propuesto una arquitectura software para la integración de CEP y SOA 2.0, utilizando un ESB como nexo de unión. Con este enfoque, CEP permitirá enviar alarmas en tiempo real cuando se produzcan situaciones relevantes a partir de la detección de ciertos patrones de eventos definidos previamente, y SOA facilitará la integración de sistemas de información heterogéneos. En las primeras etapas de investigación de esta propuesta, sobre las que versa este artículo, nos hemos centrado únicamente en el uso del motor CEP para la detección de alarmas en tiempo real.

Además, se ha descrito e implementado un caso de estudio en el que se ha aplicado esta arquitectura. Los resultados confirman que este sistema puede detectar epidemias y pandemias en tiempo real, frente a la mayoría de las herramientas actuales que informan de estas situaciones con una periodicidad semanal. Asimismo concluyen que la tecnología CEP se adecúa para dicho propósito. Aunque se ha tomado como ejemplo el virus de la gripe aviar, este sistema podrá utilizarse para la prevención de otras enfermedades, incluso en otros campos ajenos a la medicina, para lo que será necesario la definición de nuevos patrones de eventos.

En nuestro trabajo futuro más próximo compararemos el rendimiento de nuestro sistema con el de una solución más convencional que no use CEP. Asimismo, mediremos la ganancia de rendimiento que pueda suponer el uso de base de datos NoSQL con respecto a las relacionales. Por otro lado, completaremos la integración de este sistema CEP con SOA 2.0, de acuerdo con la arquitectura software propuesta, estableciendo mecanismos de seguridad e integridad para los eventos. Esto hará posible que CEP se aplique sobre eventos "reales" que serán publicados al ESB por distintos productores de eventos de distinta naturaleza y provenientes de fuentes dispares. Además, tanto los productores como los consumidores de eventos podrán ser servicios web, obteniéndose así un sistema más complejo pero, a su vez, débilmente acoplado, modular y más flexible.

Agradecimientos. Los autores agradecen a Víctor Ayllón y Juan Manuel Reina, fundadores de la empresa Novayre, las discusiones y conocimientos intercambiados sobre la aplicación de CEP en entornos SOA. El segundo autor agradece el apoyo del proyecto TIN2008-02985 y los fondos FEDER.

Referencias

1. Drools (marzo 2011), http://www.jboss.org/drools
2. Esper (abril 2011), http://esper.codehaus.org
3. EuroFlu (enero 2011), http://www.euroflu.org/index.php
4. FluNet (febrero 2011), http://www.who.int/csr/disease/influenza/influenzanetwork/flunet/en/
5. GISAID (mayo 2011), http://www.gisaid.org
6. Java VisualVM (mayo 2011), http://visualvm.java.net/
7. JBoss ESB (marzo 2011), http://jboss.org/jbossesb
8. NoSQL databases (abril 2011), http://nosql-database.org/
9. Organización Mundial de la Salud (abril 2011), http://www.who.int/es
10. Ayllón, V., Reina, J.M.: CEP/ESP: Procesamiento y Correlacion de gran Cantidad de Eventos en Arquitecturas SOA. In: 4th Jornadas Científico-Técnicas en Servicios Web y SOA. pp. 97–110. Sevilla (2008)
11. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. Wiley, Reino Unido (1996)
12. Chandy, K.M., Schulte, W.R.: Event Processing: Designing IT Systems for Agile Companies. Estados Unidos (2009)
13. Havey, M.: CEP and SOA: Six Letters Are Better than Three (2009), http://www.packtpub.com/article/cep-complex-event-processing-soa-service-oriented-architecture
14. He, M., Zheng, Z., Xue, G., Du, X.: Event Driven RFID Based Exhaust Gas Detection Services Oriented System Research. In: 4th International Conference on Wireless Communications, Networking and Mobile Computing. pp. 1–4 (2008)
15. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Estados Unidos (2001)
16. Meijer, E., Bierman, G.: A co-Relational Model of Data for Large Shared Data Banks. ACM Queue 9, 30–48 (2011)
17. Organización de las Naciones Unidas: La Gripe Aviar y la Amenaza de la Pandemia (febrero 2011), http://www.un.org/spanish/influenza/topics/pandemic_influenza.shtml
18. Sosinsky, B.: Cloud Computing Bible. Wiley, Estados Unidos (2011)
19. Sottara, D., Manservisi, A., Mello, P., Colombini, G., Luccarini, L.: A CEP-based SOA for the Management of WasteWater Treatment Plants. In: IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems. pp. 58–65 (2009)
20. Taher, Y., Fauvet, M., Dumas, M., Benslimane, D.: Using CEP Technology to Adapt Messages Exchanged by Web Services. In: 17th International Conference on World Wide Web. pp. 1231–1232. ACM, China (2008)
21. Velandia-Briñez, C., Don, S., Cho, N., Choi, E., Min, D.: Breast Cancer Image Classification Based on a Complex Event Processing Engine. In: 4th International Conference on New Trends in Information Science and Service Science. pp. 60–64 (2010)

Repositorio multiadministración

Fundació IBIT – Tel: +34 971177270/71 Fax: +34 971177279

Felip Salas. fsalas@ibit.org +34 971 177277

Rafel Barcelo. rbarcelo@ibit.org +34 971 177283

Antoni Reus. areus@ibit.org +34 971 177282

Mª Magdalena González. mgonzalez@ibit.org +34 971 177299

Esteve Schouten. eschouten@ibit.org +34 917 177270

Resumen

El repositorio multiadministración es un sistema de información capaz de agregar contenidos provenientes de sistemas heterogéneos, mediante la utilización de servicios web. Sincroniza información proveniente de fuentes de datos de terceras entidades y ofrece mecanismos de actualización con las mismas para permitir la veracidad y máxima actualidad de la información. El repositorio ofrece también herramientas para poner la información a disposición de la ciudadanía permitiendo realizar búsquedas temáticas que abarcan los recursos de todas las administraciones integradas.

En particular permite actualmente que toda la información administrativa de cualquier administración pública de las Illes Balears esté disponible en un único repositorio de datos. Este hecho facilita su posterior explotación a través de cualquiera de las canales posibles, ya sea vía web, vía telefónica, u otros canales. Actualmente el repositorio multiadministración provee información a los siguientes sistemas:

- Ventanilla Única de la directiva de servicios.
- Portal 060.
- Portal del Informador [1]
- Portal multiadministración [2]

Está previsto además estructurar la información contenida en el repositorio multiadministración vía formatos Open Data para facilitar su explotación a terceros.

1. Introducción

Dentro del marco del proyecto europeo *Regions On Line [3]* (2003-2004), el Govern Balear con la colaboración de la Fundación iBit desarrolló el repositorio administrativo ROLSAC[4], un sistema para gestionar la información administrativa del gobierno de la comunidad autónoma de las Illes Balears. Dicha información incluye la información organizativa, la información sobre procedimientos administrativos, su normativa y su tramitación, e información de divulgación (eventos, noticias de interés).

Durante el diseño de ROLSAC se puso énfasis en tres aspectos:

- Definir una estructura amplia y clara de la información administrativa a explotar en el repositorio.
- Permitir la clasificación de la información desde un punto de vista orientado a la ciudadanía, introduciendo para ello el concepto de materia para agrupar los contenidos por temáticas y el de hecho vital, que permitiese relacionar los procedimientos administrativos asociados a determinados momentos de las vidas de las personas.
- Permitir la "multi-entidad". Es decir, que una sola instancia del sistema pudiera albergar la información de diversas administraciones públicas, para permitir su uso en modo ASP (*Application Service Provider*). De esta manera se permitiría, por ejemplo, que con una sola instalación, una entidad supramunicipal pudiera dar servicio a varios ayuntamientos.

Durante los años siguientes, dentro de un grupo de trabajo de las diferentes administraciones públicas de Baleares, que se denominó ROL2 [5], se realizó un trabajo de intercambio de experiencias y de fomento de la armonización del contenido y clasificación de la información administrativa de las diferentes administraciones de Baleares, acordando una clasificación común de materias y hechos vitales.

Dentro de este grupo, que posteriormente se formalizaría como la Comisión Balear de TIC (CBTIC), surgió la iniciativa de realizar un sistema para ofrecer toda la información administrativa de las instituciones de Baleares en un mismo portal.

Para ello se usó como base el gestor ROLSAC, que ya contaba con las funcionalidades necesarias para albergar la información de varias administraciones y disponía de una estructura de información y clasificación ámpliamente aceptadas por las diferentes administraciones. Además se procedió a su liberación como software libre para fomentar su adopción entre las administraciones que no contaban con gestores de contenidos adecuados para mantener su información administrativa.

El principal trabajo realizado, objeto de esta comunicación, consiste en la definición y desarrollo de

los diferentes servicios web dentro de ROLSAC que ha permitido crear un sistema, en el cual una instancia de ROLSAC, denominado Repositorio Multiadministración, importa la información administrativa y se mantiene sincronizado con otras instancias de ROLSAC de las administraciones baleares o con sus gestores de contenidos que implementen los servicios definidos para el intercambio de información.

2. Comunicación entre sistemas

El escenario está formado por los sistemas que actúan de Administraciones Remotas, aquellas de las que se quiere obtener información y por el destinatario o destinatarios, aquellos que almacena la información recibida y que es necesario tener en todo momento actualizados.

En la siguiente figura se muestra la comunicación entre los diferentes actores. Las administraciones remotas son sistemas gestores de contenidos que implementan el servicio SincronizaciónServicio y uno de los destinatarios es el repositorio multiadministración que implementa el servicio ActualizaciónServicio. Como se puede observar en la figura, puede haber varios destinatarios. En concreto, el repositorio multiadministración se comunica con las administraciones remotas a través del ESB (*Enterprise Service Bus* [6]) para solicitarles toda la información inicial, esto se hace invocando el servicio SincronizacionServicio, es un proceso unitario y su duración depende del volumen de información a transferir. Una vez realizada la Sincronización, las diferentes administraciones Remotas actualizarán la información invocando al servicio ActualizacionServicio del destinatario. También se pueden observar los distintos canales de explotación de los datos del repositorio multiadministración.

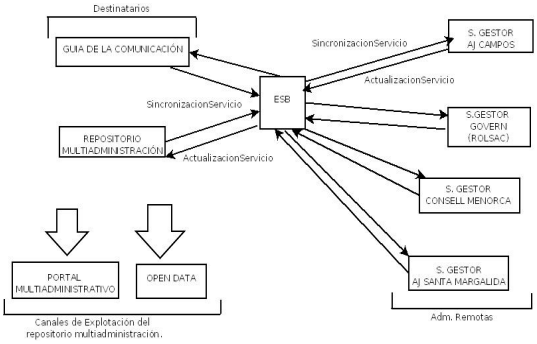


Figura 1: Comunicación entre sistemas.

Entre los diferentes componentes interconectados se introdujo un bus de servicios (en esta caso Mule ESB). La introducción de este componente no se debe a la complejidad de las rutas de los servicios, ya que en esencia se limita a ejercer de puente entre los servicios web. Sin embargo su uso ha permitido introducir un conjunto de servicios de valor añadido:

- Armonización de los puntos de acceso a los servicios.
- Sistema de bitácora de los mensajes intercambiados.
- Tolerancia a errores puntuales de red, mediante políticas de re-intento.

Su uso también facilitará la interconexión con gestores de contenidos que tengan dificultades en implementar los servicios web requeridos por el sistema, ya que provee un entorno y un conjunto de herramientas para facilitar la transformación de las peticiones o respuestas, por ejemplo mediante plantillas XSLT (*EXtensible Stylesheet Language Transformations*) o el uso de otros tecnologías de conexión como JMS (*Java Message Service*).

En el entorno de producción actual, el repositorio multiadministración contiene la información administrativa de diferentes administraciones de las Illes Balears que son el Consell de Menorca y el propio Govern y se está trabajando para incluir el Consell de Mallorca, el Consell de Eivissa y numerosos ayuntamientos.

3. Descripción de los Servicios

A continuación se describen los diferentes servicios web que permiten el intercambio de información entre los sistemas intercomunicados:

SincronizacionServicio

Permite obtener la información de una administración remota. Para ello el repositorio multiadministración o cualquier destinatario invoca los métodos que ofrecen las administraciones remotas. Esta sincronización solo se realiza una vez. Es un proceso unitario.

ActualizacionServicio

Permite a las administraciones remotas actualizar en los destinatarios la información que haya sufrido cambios en su sistema local. Para ello invoca a los distintos métodos de actualización que proporciona el servicio web.

4. Conjunto de información que se transfiere

El conjunto de información que transfieren las administraciones remotas al repositorio multiadministración consiste en :

- Unidades Administrativas
- Procedimientos Administrativos y sus trámites
- Fichas informativas
- Edificios

5. Versiones de los Servicios

Actualmente existen tres implementaciones que son fruto de la evolución del repositorio para su explotación a través de diferentes sistemas. La versión 1 es la versión que contiene las necesidades iniciales. La versión 2 es una ampliación de la información a transferir demandada por el portal multiadministración y portal del informador [1] [2] y la versión 3 corresponde a la adaptación para

integrarse con la Ventanilla Única de Directiva de Servicios [7].

Las diferencias entre versiones radican en la cantidad de información que se transfiere en cada una de ellas, haciéndose necesario el aumento de métodos específicos que implementen la nuevas funcionalidades.

La descripción de los servicios a través de su WSDL (*Web Services Description Language)* se puede consultar aquí:

http://porinf.ibit.org/sacws/services

Versión 1

El conjunto de información habilitada para ser transferida comprende Unidades Administrativas, Fichas informativas y Procedimientos asociados a una determinada Unidad Administrativa.

Versión 2

En esta versión se amplia el conjunto de información transferible de la versión 1, añadiendo los edificios asociados a una determinada unidad administrativa.

La versión 2 ofrece una ampliación de métodos con respecto a la versión 1 que son los que permiten obtener dichos edificios.

Versión 3

La versión 3 a diferencia de las dos anteriores no amplía el conjunto de información transferible, sino que se amplía la información a transferir de un procedimiento administrativo para adaptarlo a la información requerida por la Ventanilla Única de la directiva de servicios [7].

6. Experiencias prácticas

El repositorio ofrece información actual del Govern de les Illes Balears y del Consell Insular de Menorca, de modo que se tienen las experiencias prácticas de la recogida de información de ambos sistemas, así como, la actualización de los datos. La experiencia ha mostrado que el proceso de sincronización al ser un proceso unitario y dependiente del volumen de información a transferir es un proceso sensible y delicado. Con el Govern de las Illes Balears se han tenido problemas de memoria con el MULE ESB ya que no era capaz de gestionar la gran cantidad de información que le llegaba. Eso ha llevado a cambiar el proceso de sincronización para que la información que se envía a la vez sea más reducida. También se ha detectado que es un proceso dependiente de la información en sí misma, si hay alguna incongruencia en la información, el proceso automáticamente se para y es necesario borrar lo que se haya transferido y volver a iniciar el proceso. En cuanto al servicio de Actualización no se han experimentado muchos problemas debido a que la actualización se realiza de elementos simples y uno a uno, por tanto no se han experimentado problemas de memoria.

7. Conclusiones

El desarrollo de la administración electrónica, como pieza clave para la modernización de los servicios públicos, requiere que las administraciones compartan información y trabajen de manera coordinada. En este sentido la creación de un repositorio de información administrativa de interés para el ciudadano habilita múltiples posibilidades en materia de explotación de información, estudios de hábitos, estandarización informativa, facilitación de la interoperabilidad, publicación en formatos abiertos, etc. En definitiva este proyecto permite facilitar la compleja relación del ciudadano con la administración.

8. Agradecimientos

Al Govern de les Illes Balears como impulsor del proyecto ROL y en concreto a:

- la Dirección General de Tecnología y Comunicaciones
- la Dirección General de Calidad de los Servicios

A los diferentes consells insulars:

- Consell Insular de Mallorca
- Consell Insular de Menorca
- Consell Insular de Ibiza
- Consell Insular de Formentera

Referencias

[1][2] Portal multiadministración y portal del informador.
http://www.administracionsbalears.net/
[3] Web proyecto ROL(*Regions On Line*)
http://www.caib.es/rol/
[4] ROLSAC.
http://programarilliure.caib.es/sacmicrofront/contenido.do?idsite=1625&cont=28507
[5] Web proyecto ROL 2(*Regions On Line*).
http://www.caib.es/rol2/
[6] Mule ESB.
http://www.mulesoft.org/
[7] Ventanilla Unica de Directiva de Servicios.
http://www.eugo.es

Arquitectura de Negocio y Tecnológica para la Administración Electrónica

Daniel González Morales[1], José Luis Roda García[2], Pedro González Yanes[2], Soledad Montelongo Betancor[3]

[1] Agencia Canaria de Investigación, Innovación y Sociedad de la Información
dgonmor@gobiernodecanarias.org
[2] Universidad de La Laguna
{jlroda,pgonyan}@ull.es
[3] Instituto Tecnológico de Canarias
smontelongo@itccanarias.org

Abstract. Uno de los retos más importantes que tienen en este momento las administraciones públicas es la introducción de la administración electrónica. En este trabajo se presenta la forma en que la Agencia Canaria de Investigación, Innovación y Sociedad de la Información ha abordado este proyecto centrándose en la arquitectura tecnológica y funcional. Se basa en una arquitectura orientada a servicios en la que cada componente aporta servicios de un alto nivel de abstracción. Los sistemas de información que gestionan los diferentes procedimientos utilizan estos servicios siguiendo el paradigma Software Product Line y para el desarrollo de los mismos se aplica Model Driven Engineering, donde el modelo de negocio se describe mediante un Domain Specific Language.

Keywords: Administración electrónica, Arquitectura orientada a servicios, MDE, SPL, DSL.

1 Introducción

La Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI, en adelante) es el órgano de la Administración Pública de la Comunidad Autónoma de Canarias competente en el fomento de la investigación, el desarrollo científico y tecnológico, de la innovación empresarial y del despliegue de infraestructuras de telecomunicaciones y de servicios de la sociedad de la información.

Con el objetivo de adaptarse a la Ley 11/2007 [1], de Acceso Electrónico de los Ciudadanos a los Servicios Públicos (LAECSP) se redactó a principios del año 2008 el Plan Director de Tecnologías de la Información de la ACIISI, cuyo principal objetivo era dar cumplimiento a dicha ley en los plazos establecidos, el 1 de enero de 2010. Este plan definió los proyectos a desarrollar, la tecnología a emplear, la arquitectura de los sistemas de información y la forma de gestionar los proyectos.

La implantación de la administración electrónica en un organismo implica cambios legales, cambios procedimentales y organizativos, cambios en la relación con el ciudadano y la introducción de tecnología. Este trabajo se centra en la arquitectura tecnológica y funcional para adoptar la administración electrónica.

En la sección dos se describe la arquitectura de negocio y las necesidades que la tecnología debe resolver. En la sección tres se muestra la arquitectura tecnológica y se detalla la interacción de los diferentes componentes, la tecnología empleada y como resuelven las necesidades planteadas en el modelo de negocio. Por último, en la sección cuatro se describen las principales conclusiones y las actividades que se están desarrollando o que se tienen que abordar en el futuro.

2 Arquitectura de Negocio

Antes de comenzar el desarrollo de los sistemas de información se definió una arquitectura del negocio describiendo las funciones y procesos necesarios para implantar la administración electrónica. Gran parte de ellos están identificados en la LAECSP y en la normativa de desarrollo.

En mayo de 2009 se publicó el decreto 56/2009, del Presidente, por el que se regula la utilización de medios electrónicos en la Agencia Canaria de Investigación, Innovación y Sociedad de la Información.

Durante la definición de este modelo de negocio se aplicaron técnicas de simplificación y modernización administrativa que cambian de forma importante la manera de actuar respecto a la tramitación en papel. La concreción de este modelo para la gestión de subvenciones se describió en el Proceso General Electrónico de Gestión de Subvenciones.

Durante el año 2010 se modificaron todas las convocatorias de subvenciones de la ACIISI para adaptarlas a la legislación sobre administración electrónica, al decreto 56/2009 y al nuevo modelo de negocio.

Los elementos resultantes de este proceso inicial de simplificación y modernización, que componen la arquitectura de negocio que se han desarrollado son los siguientes:

Archivo electrónico de documentos.

Todos los documentos electrónicos son almacenados en el archivo electrónico de documentos (AED), y todos los sistemas de información usan este componente para almacenar los documentos. El AED está estructurado en series documentales que contienen expedientes electrónicos. Las series documentales son conjuntos de expedientes electrónicos del mismo tipo que tienen una gestión común. Los expedientes electrónicos contienen los documentos electrónicos relacionados con un asunto concreto de un afectado concreto.

Sede electrónica.

La sede electrónica (https://sede.gobcan.es/aciisi) es el punto de contacto entre la administración y el ciudadano y desde ella se puede acceder a todos los servicios electrónicos. Es el equivalente en electrónico a las sedes físicas. Una diferencia importante con los portales web es que la administración es responsable de la integridad, veracidad y actualización de la información y los servicios.

Registro electrónico de entrada y salida.

Es responsable de registrar todas las recepciones de documentación presentadas por los ciudadanos y enviar dicha información a las unidades gestoras para que los incorporen a los expediente y procedan con su tramitación.

Un elemento importante es que los ciudadanos deben tener constancia de haber presentado determinada documentación en el registro electrónico. El registro electrónico de salida deja constancias de las comunicaciones del órgano administrativo hacia el ciudadano u otras administraciones.

Portafirma electrónico.

Permite la firma de documentos electrónicos y es el equivalente a los portafirmas en papel.

Notificación electrónica.

Determinados documentos producidos por la administración deben ser notificados de forma fehaciente a los interesados. El sistema debe garantizar la integridad del documento, la autenticidad del ciudadano, la identificación del órgano, la fecha de puesta a disposición y la fecha de acceso.

Verificación de la documentación.

Consiste en comprobar que la documentación que obra en el expediente es suficiente para tomar la decisión sobre en qué sentido resolver el mismo. En el caso de que no sea suficiente se emite un requerimiento indicando al ciudadano la documentación que falta y las causas que invalidan la documentación presentada.

Baremación.

Este proceso consiste en valorar el expediente aplicando una serie de criterios que permitan tomar la decisión sobre el sentido de la resolución. En el caso de las subvenciones los expedientes se ordenan de acuerdo con la puntuación obtenida.

3 Arquitectura de la Solución Adoptada

La solución desarrollada se apoya en el paradigma de Software Product Line (SPL)[2]. La gestión de la administración pública es un claro ejemplo de SPL. Es necesario disponer de un conjunto de sistemas software, uno por cada procedimiento administrativo, que comparten un núcleo común. En el caso de la administración hay un conjunto de leyes y sus desarrollos reglamentarios que definen ese núcleo común: Ley 30/1992[3], Ley 15/1999[4] y LAECSP, entre otras. Por un lado, a pesar del ámbito de negocio tan amplio de la administración pública, las diferencias en la forma de gestionar los diferentes procedimientos es muy baja, y por el otro, al estar la parte común soportada por varias leyes el marco es muy estable en el tiempo.

Evidentemente, el dominio de aplicación es la gestión administrativa y las características deben soportar las diferencias entre los diferentes procedimientos administrativos. Los sistemas de información generados usan los activos de un conjunto de componentes software que disponen de servicios de un alto nivel de abstracción y que están orientados al negocio. Por último, existe un procedimiento de cómo definir, desarrollar e implantar un nuevo sistema de información.

Por otro lado, los diferentes sistemas de información se desarrollan bajo el paradigma de MDE[5]. Existe un modelo independiente de la plataforma

documentado en el Procedimiento General Electrónico de Gestión de Subvenciones. Sin embargo, puede ser utilizado para otros procedimientos administrativos. El modelo de dominio se describe con un DSL[6] textual.

El modelo específico de la plataforma se compone de una arquitectura orientada a servicios cuyo núcleo es un Enterprise Service Bus. Concretamente se ha utilizado la solución ServiceMix [7] para su implementación. Además, se ha desarrollado un conjunto de componentes que exponen sus funcionalidades mediante Web Services y que interactúan entre sí. En este mismo apartado se describen con mayor detalle.

Por otro lado, se utilizan servicios corporativos del Gobierno de Canarias de la plataforma Platino[8]. Ésta ofrece servicios de administración electrónica como firma electrónica, registro electrónico de entrada, registro electrónico de salida, pasarela de pagos, etc.

A lo largo de varios años, el grupo Taro de la Universidad de La Laguna ha trabajado en varios proyectos para el Servicio Canario de Empleo y la Consejería de Asuntos Sociales del Gobierno de Canarias usando siempre un enfoque MDE y herramientas de generación de código como pueden ser XML y transformaciones XSLT [9], MDA[10], AndroMDA[11], EMF[12], GMF[13], UML Profiles[14], QVT[15], Sculptor[16], Moskitt[17], etc.

El grado de éxito y los inconvenientes encontrados en la ejecución de los proyectos realizados han sido variados, ya sea bien por complejidad de la estrategia generación, bien por la complejidad de las tecnologías usadas y la madurez de las herramientas, bien por la valía de los equipos de desarrollo, etc. Esto nos ha permitido proponer el enfoque que aquí presentamos.

La herramienta de transformación que hemos desarrollado genera el código de la aplicación para la plataforma específica de dominio. La capa de presentación y de datos no pueden ser modificadas manualmente, sin embargo, los desarrolladores pueden introducir elementos específicos de un procedimiento administrativos en la capa de negocio, complementando la variabilidad que se puede expresar con el DSL. La herramienta de transformación está desarrollada sobre Open Architecture Ware. La gramática del DSL se describe XTEXT[18] y las transformaciones usan XPAND[19] y clases java.

Las aplicaciones generadas están escritas en java y se utiliza JPA para la capa persistente de acceso a base de datos, Spring Framework para la inyección de dependencias, control de transacciones en base de datos, acceso a servicios web, seguridad de la aplicación y para el control del ámbito de los objetos. Para la vistas se utiliza JSF 1.2 RichFaces que cuenta con componentes gráficos AJAX, validadores y conversores. Por último, se utiliza Openoffice y Jasper Reports para la generación de informes en PDF.

En la figura 1 se muestran los principales componentes de la arquitectura desarrollada y las relaciones entre ellos.

Los servicios que ofrece esta arquitectura y la tecnología utilizada son:

Base de datos de procedimientos.

Almacena información de configuración de cada uno de los procedimientos electrónicos. Esta información es usada por el resto de componentes para definir su comportamiento. Para cada uno de los tipos de procedimientos se deben declarar los tipos de trámite. Cada trámite puede tener tipos de documentos cuya obligación de presentación está tipificada. Cada tipo de documento contiene un conjunto de

metadatos que describen el documento, además, dispone de códigos de requerimiento. Éstos son los motivos tasados por los cuales un documento puede ser declarado no valido. Cada procedimiento tiene un conjunto de motivos de exclusión tipificados.

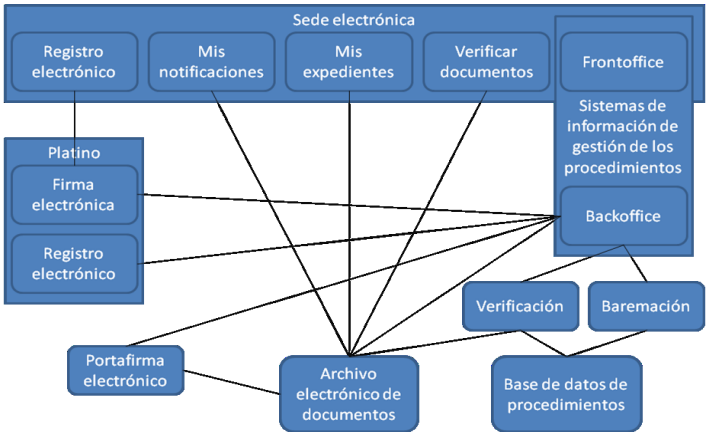


Fig. 1. Componentes principales.

Archivo electrónico de documentos.

Para gestionar el archivo electrónico de documento se ha utilizado el gestor documental Alfresco [20] sobre el que se ha desarrollado una capa de servicios que incorporan una serie de funcionalidades y exigencias de las LAECSP como la organización en series documentales y expedientes electrónicos, el foliado electrónico y la conversión de documentos firmados electrónicamente en documento con el pie de firma.

Los sistemas de información utilizan esta capa de servicio y no acceden directamente a Alfresco, lo que nos permite independizarnos de este.

Sede electrónica.

Desde el punto de vista tecnológico la sede electrónica se puede ver como un portal web que tiene incorporados un conjunto de servicios electrónicos. Como solución tecnológica hemos escogido Joomla [21] al que se le han añadido los siguientes componentes: registro electrónico, mis expedientes, mis notificaciones y verificación de los documentos.

Registro electrónico de entrada.

Es un componente de la sede electrónica que muestra un formulario que permite a los ciudadanos presentar cualquier documentación no normalizada. El componente genera un documento pdf con la solicitud al que le añade los códigos hash y las descripciones de los documentos presentados, posteriormente llama al servicio de Platino de Registro de Entrada. Este servicio devuelve el número de registro, la fecha y hora, y un recibo de la presentación realizada firmada con el sello electrónico. Por último, el componente almacena todos los documentos metainformados, incluido la solicitud y el recibo del registro, en el archivo electrónico de documentos.

Mis expedientes.

Se encarga de dar soporte a la transparencia de la gestión hacia los ciudadanos. En lugar de preguntar "¿Cómo va lo mío?" pueden acceder a sus expedientes, al estado

de los mismos y a los documentos que lo componen. Este componente después de autenticar al ciudadano consulta al archivo electrónico de documentos, a través de su capa de servicios, los expedientes y documentos en los que consta dicho ciudadano como interesado. Esta información se almacena en los metadatos de los expedientes y documentos.

Mis notificaciones.

Las notificaciones por comparecencia en sede electrónica es un medio de notificación muy efectivo que consiste en dirigir al ciudadano a la sede electrónica para que consulte el apartado "Mis notificaciones" y acceda a las mismas. Los documentos a notificar son almacenados en el archivo electrónico de documentos en el expediente correspondiente con el metadato "notificable" informado. En ese momento se registra la fecha y hora de puesta a disposición.

Una vez que el ciudadano accede al apartado "mis notificaciones" de la sede electrónica y es autenticado por el componente, éste consulta al archivo electrónico de documento sobre los documentos en los que el ciudadano consta como interesado y que están marcados con el metadato "notificable".

El componente muestra estos documentos al ciudadano y almacena la fecha y hora de acceso al mismo. Esta información es transmitida posteriormente a la aplicación gestora del procedimiento para indicarle la fecha y hora en que se ha producido la notificación y que esta pueda iniciar el cómputo de los plazos que tiene el ciudadano para responder.

Verificación de documentos.

El componente de verificación de documentos solicita la dirección URL o identificador del documento y el código seguro de verificación. Con estos datos consulta al archivo electrónico de documentos que le devuelve, en caso de verificación satisfactoria los enlaces al documento original electrónico, documento que contiene la firma y el documento compuesto con el pie de firma.

Baremación.

Utiliza los servicios de la base de datos de procedimientos para obtener la información de los criterios de evaluación y los conceptos económicos del presupuesto. Los criterios de evaluación tienen una estructura jerárquica y se pueden calcular de forma manual o automática definiendo el algoritmo correspondiente.

Los evaluadores pueden acceder a cada uno de los expedientes para cumplimentar la parte manual de la evaluación. Este componente es utilizado por los sistemas de información de gestión de las subvenciones.

Verificación de la documentación.

El componente de verificación de la documentación permite realizar el seguimiento de la documentación aportada por el ciudadano, indicar la validez de los documentos, así como, los motivos de invalidez, generar el documento de requerimiento, firmarlo, registrarlo de salida y notificarlo por comparecencia en sede electrónica.

Este componente hace uso de los servicios de la base de datos de procedimientos para obtener los documentos que se manejan en un tipo de expediente y en un trámite concreto.

Portafirma electrónico.

Se compone de una aplicación cliente que permite firma documentos electrónicos generados manualmente mediante las herramientas ofimáticas o por los sistemas de

información de gestión de los procedimientos. Para este último caso existe un conjunto de servicios web que permiten la interoperabilidad.

4 Conclusiones y Trabajos Futuros

Implantación y resultados en el negocio.

La ACIISI puso en funcionamiento la sede electrónica y el registro electrónico en mayo de 2009 y durante el año 2010 se implantaron 6 convocatorias de subvenciones desarrolladas como se ha descrito en este trabajo. Los indicadores más importantes de uso son los siguientes:

- durante el 2011 se almacenaron 33.015 documentos electrónicos en el archivo electrónico de documentos frente a los 5.581 del 2009.
- el 26,1% de la totalidad de registros de entrada de la ACIISI se realizaron en el 2010 de forma electrónica.

Ampliable y mejorable.

La arquitectura permite su ampliación incluyendo nuevos componentes y modificando el DSL y la herramienta de transformación para poderlos utilizar. También es posible mejorar los componentes existentes y permitir su utilización mediante modificaciones del DSL y la herramienta de transformación. Esto permite aprender e incorporar conocimiento nuevo al modelo de negocio.

Reutilizable por otras administraciones.

Es factible la implantación de la solución en otras administraciones y la adaptación a los servicios existentes, siempre que cumplan con una mínima capacidad de integración.

Ampliación de las funcionalidades.

Durante el 2011 se está cometiendo el desarrollo de nuevos componentes que faciliten el desarrollo de los sistemas de información como los procesos de aceptación, justificación parcial, justificación final, certificaciones, reintegros, etc.

Líneas de trabajo futuras.

Se pretende mejorar y aumentar tanto el modelo de dominio FAP como el DSL desarrollado, incluyendo más funcionalidades como la gestión económica, la gestión de resoluciones, etc.

Se debe simplificar el uso de la herramienta de generación de código. Para ello se está mejorando la documentación orientada a los desarrolladores, evaluando otros frameworks java más sencillos y creando un aplicación web que permita el uso del DSL de forma online.

Se trabajará en aumentar el nivel de expresividad del DSL incorporando elementos del lenguaje con un alto nivel semántico.

Se investigará para establecer un marco más formal orientado hacia los paradigmas SPL y MDE.

Referencias

1. Ley 11/2007. http://www.boe.es/boe/dias/2007/06/23/pdfs/A27150-27166.pdf
2. SPL. http://www.softwareproductlines.com
3. Ley 30/1992. http://www.boe.es/boe/dias/1992/11/27/pdfs/A40300-40319.pdf
4. Ley 15/1999. http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf
5. MDE. Douglas C. Schmidt. IEEE Computer, vol 39 (2), páginas 25-31. 2006.
6. DSL. France RB, Rumpe B. Domain specific modeling. Software and System Modeling 2005; 4(1):1–3.
7. Service Mix. http://servicemix.apache.org/home.html
8. Plataforma de Interoperabilidad de Gobierno de Canarias. Platino. http://www.gobiernodecanarias.org/platino
9. XSL Transformations (XSLT) Version 2.0. http://www.w3.org/TR/xslt20/. W3C Recommendation 23 January 2007.
10. Stephen J. Mellor (2004). MDA Distilled, Principles of Model Driven Architecture. Addison-Wesley Professional. ISBN 0-201-78891-8.
11 AndroMDA Framework. http://www.andromda.org/docs/index.html
12. Eclipse Modeling Framework Project (EMF). http://www.eclipse.org/modeling/emf/
13. Graphical Modeling Project (GMP). http://www.eclipse.org/modeling/gmp/
14. UML profiles http://www.uml.org/#UML2.0
15. OMG. MOF 2.0 Query/Views/Transformations. Technical Report ptc/07-07-07 Jul 2007. Available at http://www.omg.org/docs/ptc/07-07-07.pdf.
16. Sculptor Platform. http://fornax.itemis.de/confluence/display/fornax/Sculptor+%28CSC%29.
17. MOSKitt. http://www.prodevelop.es/es/productos/moskitt Visitado 2 de mayo de 2011.
18. XTEXT. http://www.eclipse.org/Xtext/
19. XPAND. http://wiki.eclipse.org/Xpand
20. Alfresco. http://www.alfresco.com/
21. Joomla. http://www.joomla.org

Construcción de un módulo de seguridad integrado en una arquitectura SOA Open Source

Víctor Ayllón, Juan Manuel Reina
NOVAYRE - www.novayre.es
C/Leonardo Da Vinci 18, 5ª Planta
Parque Tecnológico Cartuja - 41092 Sevilla - Teléfono: 954 463 108
{vayllon, jmreina}@novayre.es

Resumen. La Fundación Progreso y Salud, FPS, dependiente de la Consejería de Salud de la Junta de Andalucía, es el organismo de apoyo y gestión a la investigación en la sanidad pública andaluza. En los últimos años la FPS está encarando un proceso de definición y fortalecimiento de su arquitectura técnica basándose en una arquitectura orientada a servicios (SOA), que le permite disponer de una infraestructura de interoperabilidad entre los distintos sistemas de información que forman parte de su extenso catálogo de aplicaciones. Para poder disponer de una infraestructura SOA se necesitan un conjunto de módulos comunes y reutilizables orientados a proporcionar funcionalidad de soporte, no directamente relacionada con la lógica de negocio de las aplicaciones. Entre estos servicios transversales destacan los relativos a la seguridad de las aplicaciones, en particular los sistemas Single Sign-On (SSO). En este documento se presenta la solución SOA proporcionada por NOVAYRE para el diseño y construcción del módulo de seguridad SSO.

Keywords: SOA, SSO, CAS, Open Source, LDAP, Spring, Novayre, Web Services, JAX-WS

1 Antecedentes

Las organizaciones tienen la necesidad de simplificar el proceso de creación y desarrollo de nuevas aplicaciones, así como de fomentar la reutilización mediante el uso de servicios compartidos en varios procesos de negocio.

Las Arquitecturas Orientadas a Servicio (SOA) son un paradigma software que promueven el empleo coordinado de un conjunto de servicios reutilizables, débilmente acoplados entre sí, para dar soporte a los requerimientos de negocio.

En este contexto, la Fundación se planteó diseñar un enfoque SOA para abordar sus nuevos retos tecnológicos, entre los que se encontraba el desarrollo de un módulo de seguridad con capacidad SSO que centralizara el acceso a todas las aplicaciones. Un sistema SSO permite "transportar" la información de autenticación de un sistema a otro. Si no se implantara una solución de este tipo los usuarios estarían obligados a autenticarse en cada una de las aplicaciones del sistema. Este hecho provocaría la pérdida de tiempo en estos procesos de autenticación y también disminuiría la

"usabilidad". En lugar de esto, los sistemas SSO nos permiten centralizar los procesos de autenticación reduciendo así el número de autenticaciones.

2 Problema a resolver

El problema a resolver consiste en crear un único módulo de autenticación y seguridad con funciones SSO que centralice la administración de usuarios, en concreto las funciones de alta/baja/modificación de usuarios en el ERP corporativo, accediendo a su vez a la arquitectura SOA planteada por Novayre para resolver la problemática asociada al acceso a entidades comunes de la FPS (Centros de Trabajo, Localización, Personas, etc.)

Asimismo este módulo debería integrarse con los sistemas de autenticación de la FPS (concretamente con LDAP) facilitando considerablemente el cumplimiento de las políticas de seguridad establecidas al ser el único punto de acceso para el registro y autenticación de los usuarios.

Por último este módulo se debería construir utilizando únicamente soluciones Open Source.

3. Solución propuesta

La solución proporcionada por Novayre (figura 1) fue el desarrollo de un módulo de Seguridad que permite disponer de un control centralizado (conocido como las tres Aes):

- Autenticación
- Autorización
- Auditoría de accesos

Fig. 1. Módulo de Seguridad

El módulo fue construido únicamente utilizando soluciones Open Source, destacando Central Authentication Server - CAS para la autenticación centralizada y apoyándose en su implementación en las librerías de Spring y JAX-WS.

La arquitectura técnica diseñada es la siguiente (figura 2).

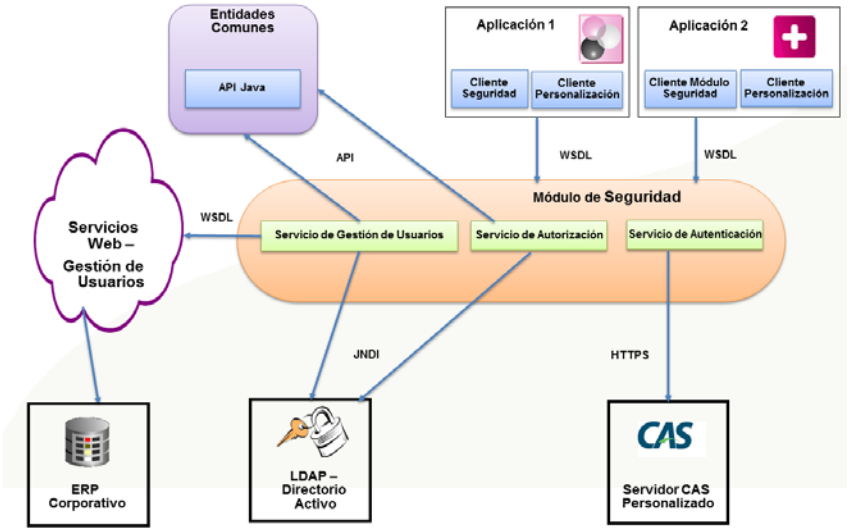


Fig. 2. Arquitectura técnica de la solución

Entre otros componentes, destacan:

Servicio de Gestión de Usuarios: permite realizar las funciones de gestión de usuarios (alta/consulta/modificación) de una forma uniforme integrándose vía Web Services con el ERP corporativo vía JNDI con LDAP

Servicio de Autorización: gestiona la autorización a las aplicaciones en base a los perfiles establecidos

Servicio de Autenticación: gestiona la autenticación única al sistema invocando a un servidor CAS personalizado y adaptado a las necesidades del proyecto. Este servidor dispone de varias páginas de login personalizadas que permiten la autenticación a través de usuario y contraseña, validando los datos en LDAP. Estas distintas versiones de las páginas de login permiten mostrar al usuario información de contexto en función de la aplicación donde se está requiriendo la autenticación, de esta forma, es posible mostrar para algunas aplicaciones que el usuario debe realizar la operación de login, mientras que en otras se muestra la página de login contextualizada para la aplicación. Este servicio también gestiona el logout único e integrado de forma que el usuario sólo debe realizar dicha operación para una aplicación propagándose este estado a las demás aplicaciones del sistema.

Aplicaciones: aplicaciones de la FPS con las que se integra el módulo de seguridad. El acceso a dicho módulo se realiza, por un lado vía un módulo Java (cliente de seguridad) desarrollado ad-hoc para facilitar la invocación de los servicios

web. Se desarrollaron dos versiones del cliente de seguridad para Java, uno para clientes legacy (Spring 2.X) y otro para clientes que utilizan Spring 3.X. Asimismo proporcionamos una configuración específica que facilita la integración de las aplicaciones Java con el sistema SSO (cliente de personalización).

Módulo de Entidades Comunes: establecido como único punto de acceso a aquellas entidades que son comunes a todas las aplicaciones de la FPS. Este módulo, construido en parte con el generador de código de Novayre - Gator™- se diseñó con una arquitectura "dual", es decir, construyendo un API Java (librería) que encapsula el acceso a los servicios para las aplicación Java y exportando a su vez los servicios vía Web Services para aplicaciones no escritas en lenguaje Java. En la siguiente imagen (figura 3) se muestra la arquitectura "dual":

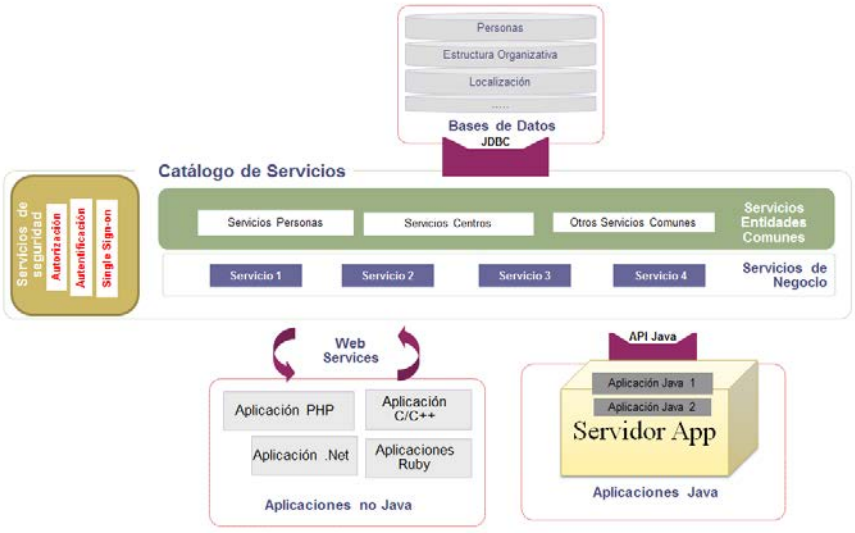


Fig. 3. Modelo de integración

Asimismo, otra consideración técnica fue la necesidad de añadir al servidor de CAS personalizado una función de "autologin" fruto de la necesidad por parte de la FPS de que existiera una página de autenticación diferente a la que proporciona CAS y que se ubica físicamente en otra aplicación web.

Esta función de "autologin" permite autenticarse en CAS proporcionando, desde una aplicación externa, las credenciales (usuario y contraseña). Además de utilizar funciones de encriptación para la comunicación con CAS, se implementó una marca de tiempo para impedir accesos no autorizados. Esta marca de tiempo implica que los links generados para el "autologin" tienen una caducidad determinada (20 segundos) lo que impide que un usuario que pueda recuperar el enlace (por ejemplo utilizando el historial del navegador) pueda acceder al sistema.

4. Conclusiones

Novayre tiene un importante reconocimiento público en la ejecución de proyectos de alto riesgo tecnológico y elevada complejidad. Durante estos años hemos conseguido sostener una diferenciación considerable a través de nuestra reputación en terminar proyectos complejos en el tiempo previsto. Nuestras capacidades nos llevan a que habitualmente se nos solicite para colaborar en proyectos de "alto impacto", por su enfoque estratégico, su visibilidad, su complejidad tecnológica o su influencia directa en resultados.

El enfoque SOA proporcionado por Novayre es un proyecto de "alto impacto" que ha permitido a la Fundación abordar los nuevos desafíos tecnológicos. En particular, la construcción de este módulo hace que los usuarios de la Fundación dispongan de un único identificador para el acceso a todas las aplicaciones corporativas y externas facilitando una experiencia de Single Sign-On (SSO) para los usuarios. Este módulo evita duplicidades en la gestión de usuarios, facilitando a su vez la reutilización de los activos existentes y la flexibilidad en los cambios. Asimismo, el disponer de un único punto de acceso a las aplicaciones aumenta la usabilidad de los sistemas ya desarrollados y como no, la propia satisfacción de sus usuarios.

Sesión 4: Ingeniería de servicios

Aplicando la Ingeniería Dirigida por Modelos para soportar la evolución de servicios

David Granada[1], Juan M. Vara[1], Vasilios Andrikopoulos[2] and Esperanza Marcos[1]

1: Grupo de Investigación Kybele, Universidad Rey Juan Carlos, Madrid, España
2: European Research Institute in Service Science (ERISS), Universidad de Tilburg, Holanda
david.granada@urjc.es, juanmanuel.vara@urjc.es, v.andrikopoulos@uvt.nl, esperanza.marcos@urjc.es

Resumen. La mejora constante de las nuevas tecnologías y los cambios en la lógica de negocio hacen que la evolución sea una característica inherente a cualquier sistema software. En el área de la orientación a servicios, la evolución es un aspecto clave dada la complejidad de actualizar sistemas distribuidos y heterogéneos. Teniendo en cuenta las anteriores premisas, en este trabajo hemos utilizado algunas de las tecnologías proporcionadas en el contexto del Eclipse Modeling Framework para proporcionar una prueba de concepto de la posible unión entre la Ingeniería Dirigida por Modelos y la Orientación a Servicios. En particular, hemos construido una primera versión de un Lenguaje Específico de Dominio para el modelado de la parte estructural de descripciones abstractas de servicios y hemos desarrollado un comparador que determina si dos versiones de un mismo servicio, expresadas con dicho lenguaje, son compatibles respecto a sus consumidores.

Palabras clave: Orientación a Servicios, Ingeniería Dirigida por Modelos, Evolución de Software.

1 Introducción

La Ingeniería Dirigida por Modelos (*Model-Driven Engineering,* MDE) es el último paso en la tendencia histórica a elevar el nivel de abstracción al que se diseña y construye el software. Los lenguajes de ensamblador dieron paso a la programación estructurada, a la que sucedió la programación orientada a objetos y así sucesivamente. Los dos principios fundamentales de la MDE son: potenciar el rol de los modelos a lo largo de todo el ciclo de vida del software y potenciar el nivel de automatización en el proceso de desarrollo [8]. Aunque los modelos siempre han sido considerados en el desarrollo de software, su uso se ha restringido tradicionalmente a tareas de documentación y en el mejor de los casos han servido para generar esqueletos del código final. Con la llegada de la MDE este escenario ha cambiado drásticamente pues el foco se traslada de las actividades de codificación a las de modelado.

En los últimos años, la MDE ha comenzado a tener una influencia directa en otras áreas. Bajo la premisa "*todo es un modelo*" [8], los investigadores y desarrolladores de esas áreas han descubierto que pueden expresar sus problemas en forma de modelos y aplicar técnicas de MDE para resolverlos o simplificarlos aprovechando la

automatización que proporcionan las herramientas disponibles. El ámbito de aplicación varía notablemente, abarcando desde dominios genéricos, como la Ingeniería Web [18] o la orientación a servicios [5] a dominios más específicos como los esquemas de base de datos [7] o la domótica [15]. Para poder expresar problemas de IS en términos de MDE son necesarios nuevos lenguajes de modelado, transformaciones de modelos, etc. En esta línea, este trabajo se centra en proporcionar métodos y lenguajes de modelado para soportar el Desarrollo Orientado a Servicios. En particular, muestra cómo herramientas y técnicas MDE se aplican para soportar un *marco para la evolución de servicios*.

La evolución de servicios es una aproximación controlada a la gestión de los cambios en un servicio y se define como *el proceso continuo de desarrollo de un servicio a través de una serie de cambios coherentes y sin ambigüedades* [1]. La evolución de un servicio se expresa a través de la creación y desmantelamiento de las diferentes *versiones de un servicio* durante su vida útil. Estas versiones deben estar alineadas entre sí de tal modo que el desarrollador pueda realizar un seguimiento de los distintos cambios introducidos en el tiempo y conocer el efecto de estos cambios sobre el servicio original. Para ser capaz de controlar el desarrollo de un servicio, el desarrollador necesita saber por qué se hizo un cambio, cuáles son sus consecuencias, y si la versión de un servicio resultante es *compatible* con los consumidores de la versión anterior. La nueva versión del servicio es compatible si sus consumidores pueden seguir utilizándolo de forma transparente.

Varias propuestas para la evolución de servicios como [4, 10] se basan en proporcionar un conjunto de pautas a seguir con el fin de asegurar la compatibilidad del servicio. Estas pautas restringen el tipo de cambios que se pueden realizar sobre la descripción de la interfaz del servicio (por lo general un documento WSDL). Sin embargo, el seguimiento de estas pautas limita la expresividad y la portabilidad a otras tecnologías, ya que depende de las características específicas de, por ejemplo, una determinada versión de WSDL. Para resolver este problema, en [1] se propone un marco teórico para la definición formal de las condiciones en que la evolución de interfaces de servicio respeta la compatibilidad del servicio. El marco propuesto se basa en el uso de modelos formales para la representación de interfaces de servicio a partir de un metamodelo común. Esta característica hace que el marco de trabajo resulte ideal para aplicar técnicas de MDE y sirve además como una prueba de concepto de la sinergia entre la MDE y la Orientación a Servicios.

El resto del artículo se estructura como sigue: la sección 2 presenta brevemente el marco formal para la evolución de servicios compatibles desarrollado en [1]. La sección 3 muestra la sinergia entre MDE y SOA presentando una primera implementación del marco teórico descrito a través de técnicas y herramientas MDE. La sección 4 utiliza un caso de estudio para validar la propuesta. Finalmente, la sección 5 analiza los trabajos relacionados y la sección 6 concluye señalando algunas direcciones para trabajos futuros.

2 Un marco para la evolución de servicios

Como ya se ha mencionado, en [1] se presenta un marco formal basado en criterios de compatibilidad de tipos y algoritmos para controlar y delimitar la evolución de servicios. El marco de trabajo extiende y aplica teorías y métodos para el control de la

evolución en lenguajes de programación orientada a objetos como la definición de subtipos y la co- y contra-varianza de entrada y salida [11, 21]. En base a estos principios, se presenta un razonador que permite decidir cuándo un cambio en la interfaz del servicio lleva a una versión compatible para los consumidores del servicio.

Más concretamente, el marco propuesto se basa en una notación independiente de la tecnología utilizada para la representación de interfaces de servicio. Dicha notación, introducida en [2] permite definir Descripciones Abstractas de Servicio (*Abstract Service Description,* ASD). Cada ASD representa una versión particular de un servicio y se puede definir como el conjunto S de todas sus versiones registradas $S = \{s_i^j\}, i = 1,\ldots,N, j \in V$, donde V es el conjunto que contiene los identificadores de versión vid para todos los registros s. S, por lo tanto, contiene una versión particular de cada uno de los registros que lo componen. Cada ASD es conforme al metamodelo que se presenta en la Fig. 1 expresado en forma de diagrama de clases. Dicho metamodelo permite modelar la interfaz del servicio de acuerdo a tres dimensiones independientes: estructural, comportamiento y propiedades no-funcionales. En este trabajo nos centramos sólo en la parte estructural.

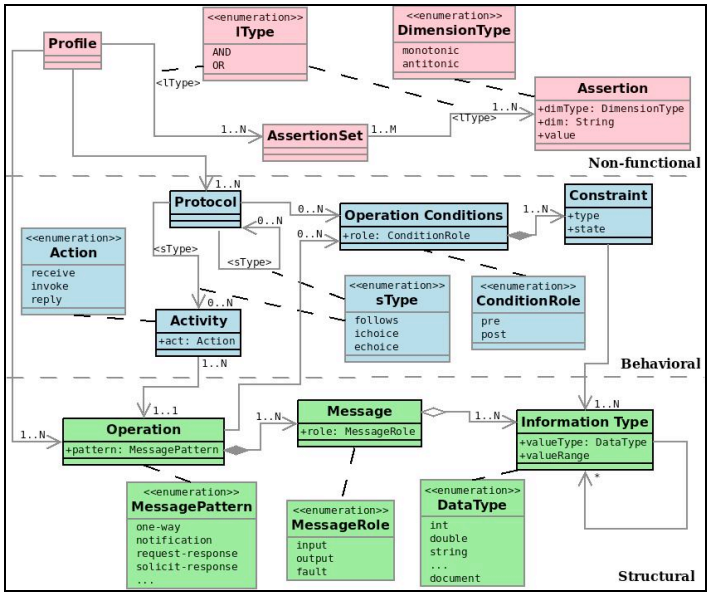


Fig. 1. Metamodelo ASD

La parte estructural del metamodelo ASD, representada en la parte inferior de la Fig. 1, permite modelar las cabeceras de los métodos y los mensajes que utilizan como parámetros que se necesitan para soportar la interacción de los clientes con el servicio. Más concretamente, esta capa contiene una clase llamada `Operation` y una clase llamada `Message`, las cuales se corresponden a los conceptos de operación y mensaje de WSDL respectivamente. `Information Type` es un contenedor para tipos complejos de esquemas XML o para elementos que son usados como parte del intercambio de mensajes. Una estructura ASD consiste en *elementos* – conceptos

informativos que representan la construcción de los bloques del servicio – y sus *relaciones* – las cuales representan las dependencias estructurales de los elementos.

Los elementos y sus relaciones son formalmente definidas como:

Definición 1. Un elemento e es una tupla $e := (name : string, (pr_{i \geq 1} : property)^*)$. Una relación $r(e_s, e_t)$ entre los elementos e_s (el elemento *origen*) y e_t (el elemento *destino*) es una tupla $r(e_s, e_t) := (name : string, name_s : string, name_t : string, rel : relation, mul : multiplicity)$ donde:

- $name, name_s, name_t$ son los identificadores únicos de los elementos e, e_s, e_t respectivamente.
- $(pr_{i \geq 1} : property)^*$ es un conjunto de cero o más *propiedades*, que toma sus valores de un *dominio de propiedades*. Por ejemplo, el tipo `messagePattern` que toma valores como `One-way`, `Request-Response`, etc. es usado para dar valor a la propiedad `pattern` de cada `operation`. or cada operación, es un ejemplo del dominio de propiedades, contiene propiedades como `One-way`, `Request-Response`, etc.
- rel es el tipo de relación entre los elementos (*a,c,s* – es decir, agregación, composición o asociación respectivamente, teniendo en cuenta la semántica definida en [2]).
- mul es la *multiplicidad* de la relación, definida como $mul := [\min, \max]$, donde $\min, \max \in \mathbb{N}$ (el conjunto de los números naturales) y representan respectivamente la multiplicidad mínima y máxima permitida para cada elemento de la relación, tal como se indica en la Fig. 1.

Los elementos e y las relaciones $r(e_s, e_t)$ son *registros* de acuerdo a la teoría de tipos definida en [11], lo cual permite definir una relación de *sub-tipificación* entre ellos:

Definición 2. Un elemento $e := (name, pr_1, \ldots, pr_k)$ es un subtipo del elemento $e' := (name', pr'_1, \ldots, pr'_m)$, y definimos $e \leq e'$, sii $name \equiv name' \wedge m \leq k \wedge pr_i \leq pr'_i$, $1 \leq i \leq m$, es decir, si los elementos tienen el mismo identificador, e' tiene menos propiedades que e y dichas propiedades son más genéricas (súper-tipos) que las correspondientes de e. Por definición se dice que $(e = \emptyset) \leq e'$.

Una relación $r(e_s, e_t)$ es un subtipo de la relación $r(e'_s, e'_t)$, y escribimos $r(e_s, e_t) \leq r(e'_s, e'_t)$, sii $e_s \leq e'_s \wedge e_t \leq e'_t \wedge rel = rel' \wedge mul \subseteq mul'$, es decir que los elementos que participan en la (nueva) relación son súper-tipos de los tipos originales y la multiplicidad de la relación es un súper-conjunto de la multiplicidad de la relación anterior. Asumimos que $(r(e_s, e_t) = \emptyset) \leq r(e'_s, e'_t)$ sii $mul' = [0, N], N \geq 1$.

Utilizando la relación de sub-tipificación y dividiendo el conjunto S de un ASD dado en dos subconjuntos S_{pro} y S_{req}, que identifican los elementos específicos de entrada y los de salida, podemos comprobar la compatibilidad de dos ASDs como sigue:

Definición 3. Dos versiones de servicios S y S' se definen compatibles, y escribimos $S <_c S'$, sii $\begin{cases} \forall s' \in S'_{req} \exists s \in S_{req} : s \leq s' & \text{(contra-varianza de entrada)} \\ \forall s \in S_{pro} \exists s' \in S'_{pro} : s' \leq s & \text{(co-varianza de salida)} \end{cases}$

El hecho que la Definición 3 pueda ser vista como una *función de comprobación de la compatibilidad*, combinada con el hecho que las ASDs pueden ser expresadas como modelos conformes a un metamodelo común, permite implementar el marco de trabajo para la evolución de la compatibilidad de los servicios aplicando técnicas de la MDE, tal como discutiremos en la siguiente sección.

3 Modelado y Comparación de ASDs

Con el fin de soportar el marco de trabajo teórico propuesto en la sección anterior, se ha desarrollado un Lenguaje Específico de Dominio (*Domain Specific Language*, DSL) para el modelado de ASDs, además de un conjunto de herramientas básico para utilizarlo. En particular, el proceso de desarrollo que hemos seguido es una adaptación del proceso propuesto en [26] para el desarrollo de nuevos entornos para el trabajo con DSLs.

El primer paso es la definición de la sintaxis abstracta del DSL, para ello de define el metamodelo en términos de Ecore utilizando las facilidades proporcionadas por EMF [15]. El siguiente paso es la definición de la sintaxis concreta del DSL. Para ello se ha utilizado EMF y GMF (Graphical Modelling Framework) [15] con el fin de generar un editor en forma de árbol y un editor gráfico o diagramador para el DSL. Finalmente, una vez definido el DSL, se ha abordado la implementación del algoritmo de comparación. Para este propósito, hemos utilizado el Epsilon Comparing Language (ECL) [19]. La siguiente sección presenta los aspectos clave de cada fase del proceso.

3.1 Definición de la Sintaxis Abstracta

La primera tarea a la hora de desarrollar un DSL es la especificación de su metamodelo. En esencia, el metamodelo recoge la sintaxis abstracta del lenguaje. En él se describen los conceptos manejados por el lenguaje y cómo estos conceptos pueden combinarse para crear nuevos modelos. El Listado 1 contiene los elementos y las relaciones presentes en la capa estructural del metamodelo ASD.

Listado 1. Especificación formal del metamodelo ASD

```
1. Operations: o := (name : string, messagePattern)
               donde messagePattern∈{one-way, notification,
               request-response, solicit-response}.
2. Messages: m := (name : string, role)
               donde role∈{input, output, fault}.
3. InformationTypes: it := (name : string, valueType, valueRange)
               donde valueType  {document, int, float, double,
               string,…} y valueRange ∈ {(min, max), N/A}.
-------------------------------------------------------------------
Relationship types:
1. Operation-Message: r(o,m) := (name_o, name_m, c, mul)
               donde mul = [1, 1..*].
2. Message-InformationType: r(m, it) := (name_m, name_it, a, mul)
               donde mul = [1, 1..*].
```

3. InformationType-InformationType: $r(it_i, it_j)$:= ($name_i$, $name_j$, s, mul) donde mul = [0, 1..*].

Para la implementación de este metamodelo, se ha utilizado Emfatic[1], un lenguaje que permite la representación textual de modelos Ecore. El uso de Emfatic permite agregar algunas anotaciones `@gmf` durante la fase de la definición del metamodelo que permiten establecer algunos aspectos gráficos del editor GMF. A partir de esta especificación se genera automáticamente un metamodelo Ecore que incluye las anotaciones `@gmf` que dirigirán la generación del diagramador. La Fig. 2 muestra dos vistas parciales de este metamodelo.

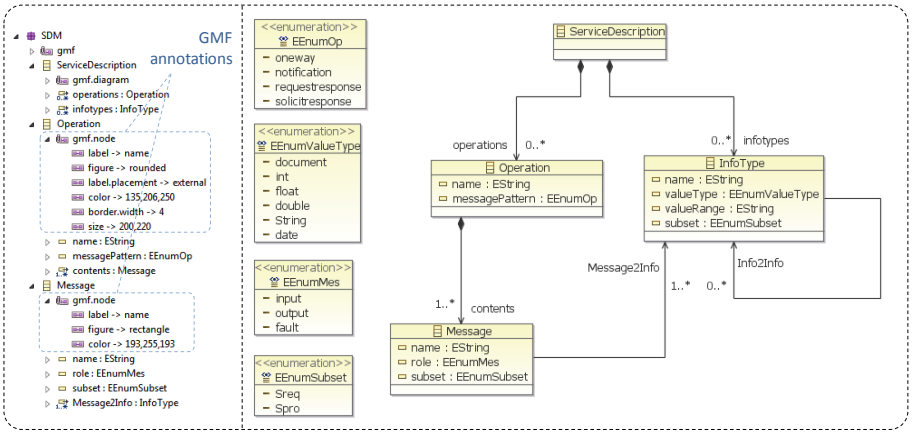


Fig. 2. Metamodelo ASD definido en términos de Ecore

La parte izquierda de la Fig. 2 muestra el editor EMF por defecto. Se puede observar como las anotaciones `@gmf` han sido asignadas a las anotaciones de tipo Ecore. La parte derecha muestra el metamodelo representado con el diagramador de Ecore, que soporta la representación de metamodelos Ecore como diagramas de clases.

3.2 Definición de la Sintaxis Concreta

Una vez definido el metamodelo del DSL que recoge la sintaxis abstracta, se debe definir su sintaxis concreta. En términos generales, la definición de la sintaxis concreta de un DSL consiste en asociar una notación a cada concepto y a cada relación del metamodelo. Nos hemos centrado en dotar al DSL de una notación visual, dado que la posibilidad de representar gráficamente cualquier modelo viene siendo una de las bases de la MDE [8]. Además, en la actualidad se trabaja para importar modelos ASD directamente desde archivos WSDL, con lo que podría utilizarse el propio WSDL como notación textual.

Como ya se ha mencionado, para el desarrollo de estos editores se utilizan las herramientas proporcionadas por EMF y GMF. Es decir, se desarrollan dos editores diferentes: uno en forma de árbol muy útil para refinar las propiedades del modelo y un diagramador, mas conveniente para ofrecer una visión general del modelo. Así, aunque encontramos que entre las herramientas para trabajar con un DSL siempre debe incluirse un diagramador, se debe considerar con mucha atención, el esfuerzo

[1] http://www.alphaworks.ibm.com/tech/emfatic

dedicado a su desarrollo, pues en realidad el editor básico en forma de árbol resulta igual de funcional. En este contexto, la naturaleza generativa de GMF encaja perfectamente: permite obtener un eficiente diagramador en un tiempo razonable y con mínimo esfuerzo. Así, el proceso seguido para proporcionar una sintaxis concreta (gráfica) al DSL para el modelado de ASD se estructura como sigue:

- A partir del (meta)modelo Ecore, EMF genera el código que implementa un editor básico en forma de árbol para modelos conformes al metamodelo de partida.
- El siguiente paso es personalizar estos editores. En [26] es posible encontrar algunos ejemplos de la personalización de editores EMF. En este sentido conviene mencionar que el código generado por GMF se basa en el código que implementa el editor EMF. Por lo tanto, las mejoras realizadas sobre los editores EMF en forma de árbol se transfieren automáticamente al editor gráfico generado con GMF.
- Finalmente, se utilizan los plug-ins EuGENia[2] y KybeleGMFgen[3] que mejoran el nivel de automatización del proceso de desarrollo dirigido por modelos implementado por GMF para la generación de editores gráficos. Este proceso mejorado consume tres entradas: el metamodelo Ecore, el código que implementa el editor en forma de árbol y un conjunto de ficheros de código Epsilon Object Language (EOL). Los ficheros EOL recogen diferentes decisiones sobre el diseño y funcionamiento del editor gráfico, ya el conjunto predefinido de anotaciones que se pueden añadir directamente sobre el metamodelo no resulta suficiente para obtener el aspecto deseado. A partir de estas entradas se generan un conjunto de modelos intermedios y finalmente el código que implementa el editor.

3.3 Comparación de ASDs

A continuación se presenta una primera versión de la implementación de la comparación entre dos versiones de un servicio para evaluar si estos son compatibles. De acuerdo con [1], podemos resumir la compatibilidad de versiones de un ASD como: *dos versiones de un servicio (representados como ASDs) son compatibles si uno de ellos está compuesto por los mismos elementos y relaciones (o un subtipo de ellos) que componen la otra versión.* Dicho con otras palabras, dos versiones de un servicio son compatibles si los elementos y relaciones de una son subtipos de los elementos y relaciones de otra. Las condiciones suficientes para que se satisfagan estas relaciones de sub-tipificación se resumen en el Listado 2.

Listado 2. Condiciones para la sub-tipificación

1. $o \leq o' \Leftrightarrow name = name' \wedge messagePattern = messagePattern'$ (se acepta sólo la igualdad).
2. $m \leq m' \Leftrightarrow name = name' \wedge role = role'$ (se acepta sólo la igualdad).
3. $it \leq it' \Leftrightarrow name = name' \wedge valueType \leq valueType' \wedge valueRange \subseteq valueRange'$

[2] http://www.eclipse.org/gmt/epsilon/doc/eugenia/
[3] http://kybelegmfgen.wordpress.com/

donde $int \leq float \leq double \leq string$ (todo lo demás no es válido) y los rangos de valores son comparados como subconjuntos (si es posible aplicarlo).

4. $r(o, m) \leq r'(o,m) \Leftrightarrow o \leq o' \wedge m \leq m' \wedge mul \subseteq mul'$.

5. $r(m, it) \leq r'(m, it) \Leftrightarrow m \leq m' \wedge it \leq it' \wedge mul \subseteq mul'$.

6. $r(it_i, it_j) \leq r'(it_i, it_j) \Leftrightarrow it_i \leq it'_i \wedge it_j \leq it'_j \wedge mul \subseteq mul'$.

7. Para todas las relaciones se mantiene que $\varnothing \leq r(s, t) \Leftrightarrow [0, 0] \subseteq mul$, o equivalentemente, si la relación tiene una multiplicidad de [0, 1..*]).

Para implementar la comparación, se traduce la especificación formal de condiciones en un programa ejecutable utilizando el Epsilon Comparison Language (ECL), un lenguaje híbrido basado en normas que permite implementar algoritmos de comparación a un alto nivel de abstracción [19]. La siguiente sección mostrará la ejecución de la comparación entre diferentes versiones de un mismo servicio.

Finalmente, conviene mencionar que el resultado del proceso de desarrollo que se ha presentado es SRMod, un plug-in de Eclipse que integra el DSL presentado, los editores para trabajar con él y la comparación implementada. Este prototipo se encuentra disponible en http://srmod.wordpress.com.

4 Caso de Estudio

Para validar el prototipo desarrollado se ha utilizado un caso de estudio tomado de la *S-Cube Network of Excellence*[4], en particular el escenario *Automotive Purchase Order Processing*, desarrollado y utilizado como uno de los escenarios de validación en dicha red. El escenario está basado en el modelo llamado *Supply Chain Operations Reference* (SCOR), que proporciona directrices para la implementación de cadenas de suministro. El escenario en cuestión, que es parte de los procesos de una compañía de la industria de la automoción, es un ejemplo de cómo implementar actividades de nivel 3 de SCOR utilizando SOA.

Se ha utilizado el prototipo SRMod para modelar dos versiones diferentes de un mismo servicio: `Receive Purchase Order`. El servicio contiene una operación que contiene dos mensajes (`POMessage` y `m2`) y utiliza un conjunto de tipos de datos (`PODocument`). Cada versión se diferencia de la otra en la cabecera de las operaciones. La primera versión del servicio considerado se muestra en el Listado 3.

Listado 3. Especificación formal del Servicio `Receive Purchase Order` (V1)

```
o1 = (processOrder, request - response)
m1 = (POMessage, input)
m2 = (m2, output) //for unnamed output message
it1 = (PODocument, document, N/A)
it2 = (OrderInfo, string, N/A)
it3 = (DeliveryInfo, string, N/A)
it4 = (it4, string, N/A) //for unnamed output message
r(o1, m1) = (processOrder, POMessage, c, [1, 1])
```

[4] http://www.s-cube-network.eu/

```
r(o1, m2) = (processOrder, m2, c, [1, 1])
r(m1, it1) = (POMessage, PODocument, a, [1, 1])
r(m2, it4) = (m2, it4, a, [1, 1])
r(it1, it2) = (PODocument, OrderInfo, s, [1, 1])
r(it1, it3) = (PODocument, DeliveryInfo, s, [0, 1])
```

En la Fig. 3 se muestra la representación gráfica de la descripción del servicio y algunas capturas de pantalla que muestran las propiedades de cada objeto.

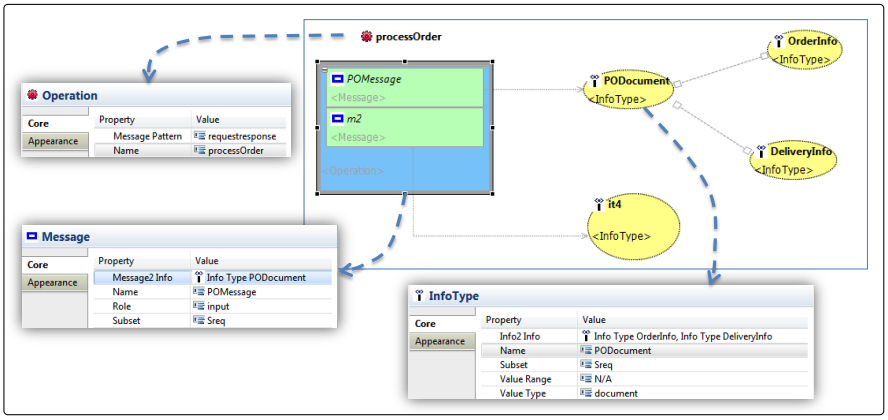


Fig. 3. Representación gráfica del Servicio `Receive Purchase Order` (V1)

La segunda versión del servicio `Receive Purchase Order` contiene un nuevo objeto del tipo `InfoType` destinado a modelar la fecha de la orden de compra (`OrderDate`), además de una nueva relación, que enlaza una orden de compra con su fecha de expedición. La nueva versión del servicio se muestra descrita formalmente en el Listado 4 (los nuevos elementos están resaltados en negrita).

Listado 4. Especificación formal del servicio `Receive Purchase Order` (V2)

```
o1 = (processOrder, request - response)
m1 = (POMessage, input)
m2 = (m2, output) //for unnamed output message
it1 = (PODocument, document, N/A)
it2 = (OrderInfo, string, N/A)
it3 = (DeliveryInfo, string, N/A)
it4 = (it4, string, N/A) //for unnamed output message
it5 = (OrderDate, date, N/A)
r(o1, m1) = (processOrder, POMessage, c, [1, 1])
r(o1, m2) = (processOrder, m2, c, [1, 1])
r(m1, it1) = (POMessage, PODocument, a, [1, 1])
r(m2, it4) = (m2, it4, a, [1, 1])
r(it1, it2) = (PODocument, OrderInfo, s, [1, 1])
r(it1, it3) = (PODocument, DeliveryInfo, s, [0, 1])
r(it1, it5) = (PODocument, OrderDate, s, [0, 1])
```

Una vez ejecutada la comparación tomando como entrada las dos versiones del servicio, los resultados se muestran en la consola de Eclipse. Actualmente se trabaja para volcar estos resultados en un modelo que luego se procesaría aplicando técnicas de MDE.

Los resultados se dividen en dos grandes grupos de acuerdo con los tipos de elementos del modelo. Así, cada grupo informa respectivamente sobre la compatibilidad entre los objetos (instancias de las metaclases recogidas en el metamodelo) y las relaciones (instancias de las meta-asociaciones recogidas en el metamodelo). Para cada elemento del modelo encontrado en una versión (elemento o relación) se comprueba si la otra versión del servicio contiene un elemento cuyas propiedades cumplan los requisitos para reemplazar al primero sin afectar a la compatibilidad del servicio.

La parte inferior de la Fig. 4 muestra que todas las comparaciones parciales presentan un resultado positivo cuando V1 se utiliza como la versión antigua del servicio, es decir, cuando V1 se sustituye por V2. De hecho, a pesar de que el tipo de dato `PODocument` utiliza un tipo de dato adicional (`OrderDate`) en V2, es igualmente posible reemplazar V1 por V2 sin afectar la compatibilidad, ya que podemos ver `V1.PODcoument` como un subtipo de `V2.PODocument`.

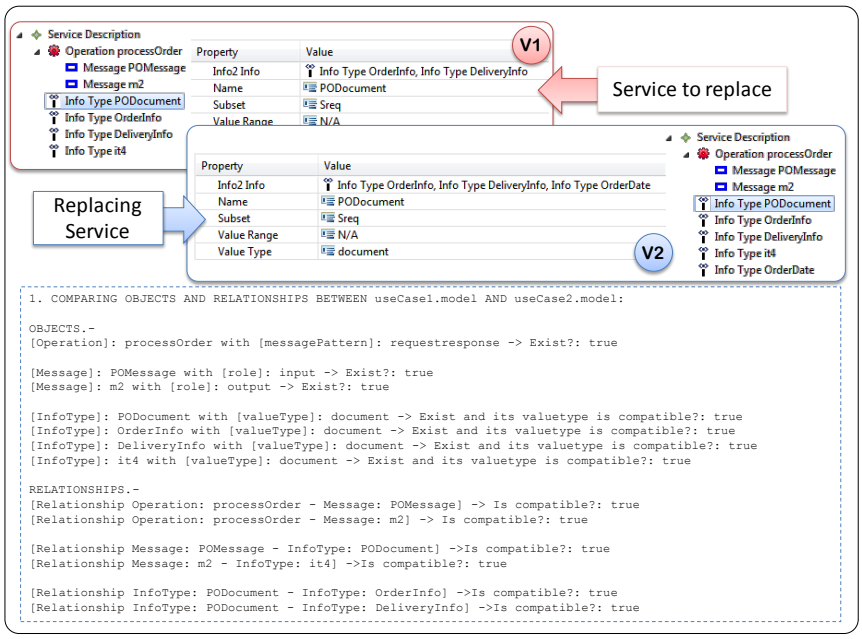


Fig. 4. Extracto del resultado de la comparación: V1 usado como versión antigua

Por el contrario, si V2 se utiliza como el servicio que se sustituye, la comparación arroja un resultado negativo ya que las relaciones de subtipo no se cumplen: no se puede sustituir el tipo de dato `V2.PODocument` por el tipo de dato `V1.PODocument`.

En resumen, el prototipo SRMod evita, en primer lugar, tener que parsear dos ficheros WSDL para buscar las diferencias entre ellos. Además, permite expresar dichos servicios de forma gráfica e intuitiva para facilitar su entendimiento; implementa el algoritmo de comparación descrito formalmente en la Sección 2 y permite expresar las diferencias de forma entendible y usable por el diseñador de los servicios.

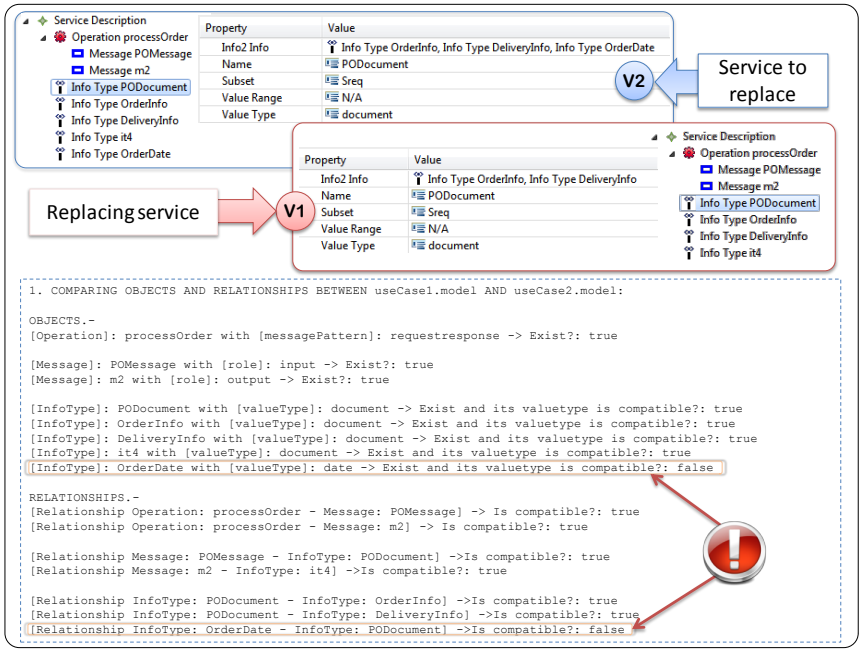


Fig. 5. Extracto del resultado de la comparación: V2 usado como versión anterior

5 Trabajos relacionados

Los procedimientos convencionales de mantenimiento (detener el sistema, modificar el código fuente y volver a arrancar el sistema) no son aplicables en entornos orientados a servicios [6]. Identificar cuáles son los componentes software que constituyen el sistema no es una tarea trivial, especialmente en el contexto de grandes redes de servicios. Además, la imposibilidad de acceder al código fuente (si existe) de los servicios proporcionados por terceras partes (principios de encapsulación y débil acoplamiento en SOA) no permite la aplicación de muchas técnicas de mantenimiento como la refactorización o el análisis del impacto, lo que ha dado lugar a diferentes enfoques para la evolución de los servicios.

En un extremo del espectro encontramos propuestas que no tienen en cuenta si los cambios introducidos por una nueva versión del servicio invalidan a sus consumidores (*break consumers*) y prefieren adoptar una aproximación lo más neutral posible [17, 20]. En el otro extremo encontramos aquellas propuestas que tienen por objetivo efectuar los cambios sin invalidar a sus consumidores de manera que las distintas versiones están contenidas dentro de la versión activa (es decir, la versión desplegada y en ejecución).

La mayor parte de las propuestas existentes se encuentran en algún punto entre estos dos enfoques [4, 10, 14, 27]. En principio, estas propuestas siguen una estrategia común orientada a soportar la evolución de servicios sin romper la compatibilidad: mantienen activas varias versiones del servicio para las entregas principales y agrupan las entregas menores en la más reciente para reducir los costes de mantenimiento. Sin embargo, estos trabajos carecen de un marco teórico que respalde las ideas prácticas

en que se basan. Por contra, el objetivo de la propuesta que se presenta es proporcionar un marco teórico y tecnológico para soportar la evolución de servicios. Este marco de trabajo funciona bajo una base formal, centrada en producir nuevas versiones compatibles de un servicio.

Respecto a la tarea de encontrar diferencias entre varios modelos, conviene mencionar los trabajos de Bernstein et al. [7] centrados en encontrar correspondencias entre esquemas de datos. En el campo de la MDE existen varios trabajos centrados en identificar las diferencias entre varios modelos UML, como [23]. En esta línea, el trabajo de Sriplakich et al. [25] eleva el nivel de abstracción, ya que permite identificar las diferencias entre modelos conformes a cualquier metamodelo. En el contexto de EMF, EMF Compare[5] es probablemente la herramienta mas aceptada para la detección de cambios entre distintas versiones de un modelo. De hecho, esta herramienta es la base para propuestas más elaboradas en torno a la evolución de metamodelos y co-evolución de modelos, tal como se muestra en [12]. Sin embargo, todas estas herramientas están orientadas hacia la identificación automática de equivalencias y diferencias. Dado que en nuestro caso necesitábamos implementar un algoritmo de comparación ad-hoc, se ha optado por utilizar el lenguaje ECL, ya que está específicamente diseñado para este tipo de tareas.

En cuanto a la convergencia entre MDE y SOA, algunos autores sostienen que la aplicación de técnicas MDE en entornos SOA, unido al uso de mejores técnicas para la documentación y mejora de los procesos de negocio, permitirán desarrollos más rápidos y ajustados a los requisitos de partida [27]. Esta alineación entre especificaciones de negocio a alto nivel e implementaciones SOA de bajo nivel, es un aspecto fundamental en el campo del desarrollo orientado a servicios. Además, durante los últimos años han surgido numerosas propuestas metodológicas para aprovechar las ventajas de combinar MDE y SOA [3, 5, 11]. Dichas propuestas proporcionan modelos, métodos y técnicas para hacer frente al desarrollo de sistemas orientados a servicios. Sin embargo, la mayoría de estas propuestas giran en torno al modelado de proceso de negocio, y por tanto a un alto nivel de abstracción. La propuesta presentada en este trabajo se aplica a nivel de servicio, lo que la hace más accesible en la práctica para los desarrolladores de servicios.

6 Conclusiones y trabajos futuros

El trabajo presentado es el primer paso en el desarrollo de un marco completo para soportar la evolución de servicios dirigida por modelos y proporciona una prueba de concepto sobre la aplicación de tecnologías MDE en entornos SOA. En particular, se han utilizado varias tecnologías MDE para implementar un conjunto de herramientas que permitan describir y versionar un servicio y analizar la compatibilidad entre versiones. Así, se ha desarrollado un DSL para modelar descripciones abstractas de servicios (ASDs) y se han construido editores gráficos para trabajar con dicho DSL. Igualmente, se ha construido un comparador básico para evaluar la compatibilidad de dos versiones de un servicio descritas con el DSL desarrollado. Para ello, se ha utilizado la noción de compatibilidad de servicios descrita formalmente en [1].

[5] http://wiki.eclipse.org/index.php/EMF_Compare

Este trabajo plantea varias direcciones para trabajo futuro: en primer lugar, la mejora de artefactos software producidos. Así, se está ampliando el DSL para cubrir no sólo la capa estructural del metamodelo ASD, si no también la capa no-funcional (parte superior de la Fig. 1). Además, se está construyendo un conjunto de extractores e inyectores para conectar el DSL para el modelado de ASDs con los lenguajes existentes para la descripción de servicios, como ejemplo WSDL. De esta forma se podrían obtener modelos ASD a partir de documentos WSDL y viceversa. Igualmente, se pretende aplicar técnicas MDE para procesar el resultado del proceso de comparación. La idea es recoger el resultado en un *modelo de weaving* que relacione los dos modelos de entrada. De esta manera, cada elemento del modelo de weaving informará sobre la compatibilidad entre dos elementos de los ASDs comparados, proporcionando una descripción básica de los posibles problemas detectados.

Agradecimientos

Esta investigación ha sido llevada a cabo en el marco de trabajo de los proyectos MODEL-CAOS (TIN2008-03582/TIN) y CONSOLIDER (CSD2007-0022, INGENIO 2010), los cuales son financiados por el Ministerio de Ciencia e Innovación de España (TIN2005-00010/), y ha recibido la financiación del séptimo European Community's Framework Programme FP7/2007-2013 en virtud del acuerdo de subvención 215483 (S-Cube).

Referencias

1. Andrikopoulos, V. (2010). *A Theory and Model for the Evolution of Software Services*. Tilburg: Tilburg University Press. Available: http://arno.uvt.nl/show.cgi?fid=107815
2. Andrikopoulos, V., Benbernou, S., and Papazoglou, M. (2008). Managing the evolution of service specifications. In *Proceedings of the International Conference on Advanced Information Systems Engineering* (CAiSE'08), Springer-Verlag, pp. 359-374.
3. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., and Holley, K. (2008). SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, *47*(3), pp. 377-396.
4. Becker, K., Lopes, A., Milojicic, D. S., Pruyne, J., and Singhal, S. (2008). Automatically determining compatibility of evolving services. In *Proceedings of the IEEE International Conference on Web Services* (ICWS 2008), pp. 161-168.
5. Bell, M. (2008). Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture, Wiley.
6. Bennett, K.H. and Rajlich, V.T. (2007). Software maintenance and evolution: a roadmap. In *Proceedings of the International Conference on The Future of Software Engineering* (ICSE), ACM, pp. 73-87.
7. Bernstein, P A. (2003). Applying Model Management to Classical Meta Data Problems. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA.
8. Bézivin, J. (2004). In search of a Basic Principle for Model Driven Engineering. *Novatica/Upgrade, V*(2), pp. 21-24.
9. Bézivin, J. (2005). On the Unification Power of Models. *Software and Systems Modeling, 4*(2), pp. 171-188.
10. Brown, K. and Ellis, M. (2004). Best practices for web services versioning. IBM developerWorks. Available: http://www.ibm.com/developerworks/webservices/library/ws-version/

11. Cardelli, L. (1988). A semantics of multiple inheritance. *Information and Computation, 76*(2-3), pp.138-164.
12. Cicchetti, A., Ruscio, D., and Pierantonio, A. (2009). Managing Dependent Changes in Coupled Evolution. In *2nd international Conference on theory and Practice of Model Transformations* (Zurich, Switzerland, June 29 - 30, 2009). LNCS, vol. 5563. Springer-Verlag, Berlin, Heidelberg, pp. 35-51.
13. De Castro, V., Marcos, E. and Wieringa, R. (2009) Towards a Service-oriented MDA-Based Approach to the Alignment of Business Processes with it Systems: From the Business Model to a WS Composition Model. *Int. Journal on Cooperative Systems*, *18*(2), pp. 225-260.
14. Fang, R., Lam, L., Fong, L., Frank, D., Vignola, C., Chen, Y. and Du, N. (2007). A version-aware approach for web service directory. In *Proceedings of the IEEE International Conference on Web Services* (ICWS'07), Jul. 2007, pp. 406-413.
15. Gronback, R. C. (2009). *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*: Addison-Wesley Professional.
16. Jimenez, M., Rosique, F., Sanchez, P., Alvarez, B., & Iborra, A. (2009). Habitation: A Domain-Specific Language for Home Automation. *IEEE Software, 26*(4), pp. 30-38.
17. Juric, M.B., Sasa, A. Brumen, B. and Rozman, I. (2009). WSDL and UDDI extensions for version support in web services. *Journal of Systems and Software*, *82*(8), pp. 1326-1343.
18. Koch, N, Meliá, S. Moreno, N. Pelechano, V. Sanchez, F. & Vara, J.M. (2008). Model-Driven Web Engineering, *Upgrade Journal, IX* (2), pp. 40-46.
19. Kolovos, D.S. (2009). Establishing Correspondences between Models with the Epsilon Comparison Language. In *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications* (ECMDA-FA '09). Springer-Verlag, Berlin, Heidelberg, pp. 146-157.
20. Leitner, P., Michlmayr, A., Rosenberg, F. and Dustdar, S. (2008). End-to-End versioning support for web services. In *IEEE International Conference on Services Computing* (ICWS'08), vol. 1, pp. 59-66.
21. B. Meyer, Object-Oriented Software Construction, 2nd ed. (1997). Upper Saddle River, NJ, USA: Prentice Hall PTR.
22. Miller, J., Mukerji, J. (Eds.), *MDA Guide Version 1.0*, OMG Document - omg/2003-05-01.
23. Ohst, D., Welle, M., Kelter, U. (2003). Differences between versions of UML diagrams. *SIGSOFT Softw. Eng. Notes, 28*(5), pp. 227-236
24. Paige, R.F., Kolovos, D.S., Rose, L.M., Drivalos, N. and Polack, F. (2009). The Design of a Conceptual Framework and Technical Infrastructure for Model Management Language Engineering. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems* (IECCS), pp. 162-171.
25. Sriplakich, P., Blanc, X., Gervais, M.P. (2006). Supporting collaborative development in an open MDA environment. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance* (ICSM'06), IEEE Computer Society, pp. 244-253.
26. Vara, J.M. 2009. M2DAT: a technical solution for Model-Driven Development of Web Information Systems. PhD Thesis. University Rey Juan Carlos. Available: http://www.kybele.etsii.urjc.es/members/jmvara/Thesis/.
27. Watson, A. (2008). A Brief History of MDA, *Upgrade*, IX, 2 (2008), pp. 7-11
28. Weinreich, R., Ziebermayr, T. and Draheim, D. (2007). A versioning model for enterprise services. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops* (AINAW'07), vol. 2, pp. 570-575.

Marco para la Modernización de Sistemas de Información basado en Modelos: Aplicación a un Caso Práctico (transmisión de *dominios .es*)

Jorge Moratalla Collado[1], Valeria de Castro[2], Marcos López Sanz[2], Esperanza Marcos[2]

[1]Entidad Pública Empresarial Red.es
(Ministerio de Industria Turismo y Comercio - Gobierno de España)
Madrid-ESPAÑA
jorge.moratalla@red.es

[2]Grupo de Investigación KYBELE
Universidad Rey Juan Carlos
Móstoles (Madrid-ESPAÑA)
{valeria.decastro, marcos.lopez, esperanza.marcos}@urjc.es

Abstract. En la actualidad la modernización de sistemas de información, entendida como la evolución tecnológica y/o funcional de un sistema existente por necesidades del negocio, se ha convertido en un factor clave para conseguir sistemas robustos, menos acoplados, con menores costes. Si tenemos en cuenta que la mayoría de sistemas no se construyen desde cero, utilizar un enfoque correcto en este tipo de proyectos se considera un factor clave para el éxito. En este contexto, *MDE (Model Driven Engineering)* se constituye como uno de los principales enfoques a la hora de abordar proyectos de este tipo. Alinear el negocio y la tecnología es fundamental para garantizar la consecución del objetivo de modernización. Este trabajo presenta una propuesta para la modernización basada en modelos, que viene apoyada por un caso de estudio real, el de la transmisión de los dominios .es.

Keywords: *Modernización de Sistemas, Knowledge Discovery Metamodel, Model Driven Development, Model Driven Architecture*

1 Introducción

En los últimos años el enfoque de desarrollo dirigido por modelos (*MDE*, *Model Driven Engineering*) se ha consolidado como una apuesta importante, que cada vez cobra más protagonismo en el desarrollo de sistemas de información.

Entre las múltiples aproximaciones de desarrollo dirigido por modelos destaca, por su amplia aceptación, la propuesta MDA [1] del consorcio OMG. MDA plantea una separación de modelos en diferentes niveles de abstracción que abarca desde el modelado de los aspectos cercanos al negocio hasta aquellos que recogen las características tecnológicas de implementación. La principal aportación de MDA al

desarrollo de sistemas no es tanto esta clasificación de modelos en distintos niveles de abstracción, sino la posibilidad de definir transformaciones (semi)automáticas entre ellos.

Por otro lado, la externalización y fragmentación del proceso industrial ha favorecido la adopción de modelos de negocios basados en *servicios*, lo que ha evidenciado la necesidad de desarrollar sistemas software cuya estructura y comportamiento es susceptible de sufrir cambios. Desde el punto de vista tecnológico, utilizando los principios del paradigma orientado a servicios (SOC, *Service Oriented Computing*), es posible crear sistemas débilmente acoplados, dinámicos y que se adaptan a las nuevas necesidades de negocio. Sin embargo, la correspondencia entre los servicios de alto nivel y su implementación mediante las tecnologías de servicios actuales no es sencilla y necesita de un proceso de alineación que dista de ser trivial. En este sentido, la utilización del enfoque MDA ayuda a solventar el salto desde las especificaciones de servicios de negocio hasta la obtención del sistema final.

A pesar de que este enfoque parece ser el más adecuado, hay que tener en cuenta que la mayoría de sistemas empresariales suelen ser complejos, heterogéneos y enfocados a una misión concreta, lo que provoca que estén fuertemente ligados a unos determinados flujos de negocio. De hecho, en el mundo empresarial raramente se construyen sistemas partiendo de cero (procesos de negocio *hardcodeados*, sin tener en cuenta la independencia funcional que debe regir el diseño de un sistema), lo que se convierte en una desventaja a la hora de rediseñarlos, o integrarlos en otros flujos más complejos. La mayoría de estos sistemas se convierten en sistemas de información heredados (*Legacy Systems*), debido a los años de mantenimiento, que suelen provocar la acumulación de parches, degradando la arquitectura del mismo e impactando en altos costes de mantenimiento.

Partiendo de esta base, y siguiendo el enfoque orientado a modelos, el OMG ha establecido un conjunto de soluciones para la modernización de sistemas de información [2], denominado ADM (*Architecture Driven Modernization)*, basado en la utilización de modelos y la aplicación de reglas de transformación entre ellos.

A pesar de que este enfoque es relativamente novedoso, existen en la actualidad algunos trabajos que promulgan la aplicación de esta estrategia de en uno o varios de sus puntos. Sin embargo, la mayoría de ellos se presentan desde una visión puramente tecnológica, o no abordan el proceso completo, quedándose en fases puntuales del mismo. Teniendo en cuenta que la alineación entre negocio y tecnología es un punto clave de cara a conseguir el éxito del proyecto, es fácil deducir que la estrategia de la empresa juega un papel fundamental como hilo conductor del mismo. Sin embargo este aspecto apenas se toca en los trabajos revisados.

Otro de los denominadores comunes en la mayoría de propuestas de modernización actuales denota un enfoque más bien teórico, o no contemplan validaciones aplicadas a casos prácticos que permitan validar el grado de viabilidad del proceso.

El objetivo del presente trabajo consiste, por lo tanto, en proponer un enfoque de modernización de sistemas de información (y en particular de sistemas legados) basado en modelos, tomando como base la estrategia ADM, de cara a plantear un marco de trabajo basado en modelos, guiado por la estrategia de la empresa, y que minimice el *gap* existente entre negocio y tecnología.

Este marco se planteará desde un enfoque práctico, abordando los modelos y las reglas de transformación en cada una de las fases desde un punto de vista genérico,

que permita obtener las reglas de negocio básicas, que serán el punto de partida para exponer servicios de valor para la organización, en un enfoque SOC.

Para validar este enfoque, la propuesta planteada será complementada con la aplicación a un caso de estudio real, el de la transmisión de nombres de dominio .es, gestionada por Red.es, Entidad Pública Empresarial adscrita al Ministerio de Industria, Turismo y Comercio (MITyC). La aplicación a este caso de estudio abordará las fases necesarias para llegar desde la situación inicial a la situación objetivo, pasando por todos los modelos y reglas necesarios para conseguirlo. Cabe destacar, no obstante, que el alcance del presente trabajo se ciñe al planteamiento del marco para la modernización de sistemas basado en modelos, y la definición de los pasos y reglas de transformación básicas entre ellos, quedando fuera del mismo la implementación de las mismas, que será propuesta como línea de continuación y sobre la que los autores ya están trabajando en otros ámbitos.

El trabajo queda estructurado de la siguiente forma. En el punto 2 se describirán los antecedentes que se han tomado como base, para luego entrar en más detalle sobre los aspectos de modernización objeto de la propuesta. El punto 3 expone el enfoque para la modernización propuesto que, posteriormente, en el punto 4, será puesto en práctica a través del caso de estudio comentado.

2 Antecedentes

2.1 Trabajos relacionados

El término modernización, unido al desarrollo dirigido por modelos, ha dado lugar al enfoque *Model Driven Reengineering (MRE)* [22] para el caso de la reingeniería, que ha sido tratado en diversos trabajos. Así, por ejemplo, Favre, J. introduce en [6] el término megamodelo como pieza fundamental para definir una ontología de modelos, pero sin detallar claramente la relación entre ellos o las reglas necesarias para transformarlos.

En parte esta carencia queda cubierta en Pu et al [5], que proponen en su trabajo una serie de reglas para obtener modelos de forma semiautomática, basados en diagramas UML, para describir las operaciones específicas del dominio. Sin embargo este tipo de transformaciones sólo se plantean desde el punto de vista de la capa web de las aplicaciones.

En otra línea, Pérez-Castillo, R. et al [8] o Cánovas J. [9] proponen marcos basados en reglas de transformación a través de lenguajes, también para la lógica del sistema. Sin embargo no contemplan la capa de negocio, o ésta se plantea desde un punto de vista más orientado al flujo que a las reglas, lo cual dificulta la posterior identificación de servicios, para su publicación, siguiendo un enfoque SOC.

Ulrich, W. da un paso más en [7] definiendo una serie de escenarios basados en los distintos tipos de modernización ADM a través de casos prácticos. Sin embargo este trabajo no define una metodología clara que guíe este proceso, al contrario que las propuestas de van der Heuvel y Chung, que presentan en [12] y [4] respectivamente varios enfoques que pueden abordarse a la hora de utilizar las distintas técnicas que se

ofrecen en la modernización de sistemas desde el punto de vista metodológico. A pesar de ello este enfoque es puramente técnico, sin entrar en detalle de cómo se podrían implementar las reglas de negocio, y que otros trabajos sí tratan, como por ejemplo el de Baxter y Hendryx, que propone en [11] la utilización de distintas herramientas y mecanismos formales para extraer la funcionalidad y las reglas de negocio, ya sea directamente a partir del código fuente existente, o de los actores participantes de la Organización, lo cual es bastante útil en los procesos de Reingeniería. Sin embargo este enfoque no plantea un enfoque de modernización basado en modelos.

Otro de los conceptos que no hay que descuidar es el riesgo que entraña cualquier proceso de Reingeniería. El trabajo de Rajala [13] ofrece algunas pautas y normas básicas que conviene tener en cuenta en cualquier enfoque metodológico, a la hora de utilizar ADM como estrategia para modernizar el Sistema existente, como por ejemplo, establecer indicadores de calidad que dirijan el proceso.

2.2 Modernización basada en ADM

Como ya se ha introducido, la iniciativa ADM define una serie de metamodelos basados en estándares, y una serie de transformaciones entre ellos para facilitar la representación de la solución para cada uno de estos dominios. ADM se apoya en el estándar MOF (*Meta Object Facility*) [17] para describir conceptos de la realidad de forma gráfica. Los metamodelos contemplados en ADM, descritos en la figura 1, son:

1. *Abstract Syntax Tree Metamodel* (ASTM), que representa la vista más granular de la Arquitectura Tecnológica del Sistema y tiene como principal objetivo soportar la transformación del mismo desde el punto de vista de los lenguajes y plataformas existentes. Este metamodelo que es descrito por el OMG en [19], podría entenderse como parte del nivel PIM de un proceso de desarrollo basado en MDA.
2. *Knowledge Discovery Metamodel* (*KDM*), descrito dentro de la iniciativa ADM del OMG en [3], representa los aspectos de la solución tecnológica y la estructura lógica del sistema, y tiene como objetivo facilitar la interoperabilidad para cualquier herramienta que dé soporte a la modernización a este nivel. Este metamodelo podría entenderse como parte del nivel PDM de un proceso de desarrollo basado en MDA.
3. *Structured Metrics Metamodel* (SMM), que es descrito por el OMG en [20], y que define una serie de metamodelos que dan soporte a la representación de métricas necesarias en el proceso de modernización.
4. *Semantic for Business Vocabulary and Rules (SBVR*): si bien la iniciativa ADM no define ningún metamodelo para representar la capa de negocio, sí que establece determinados criterios para obtener un mapeo entre los modelos representados en KDM y otro de los estándares definidos por el OMG, el SBVR, definido en [10], cuyo objetivo permite definir la semántica y las reglas de negocio asociadas a la estrategia de la Empresa. SBVR, que podría entenderse como parte del nivel CIM en un proceso de desarrollo MDA, permite por tanto desarrollar modelos semánticos basados en el vocabulario y las reglas del negocio.

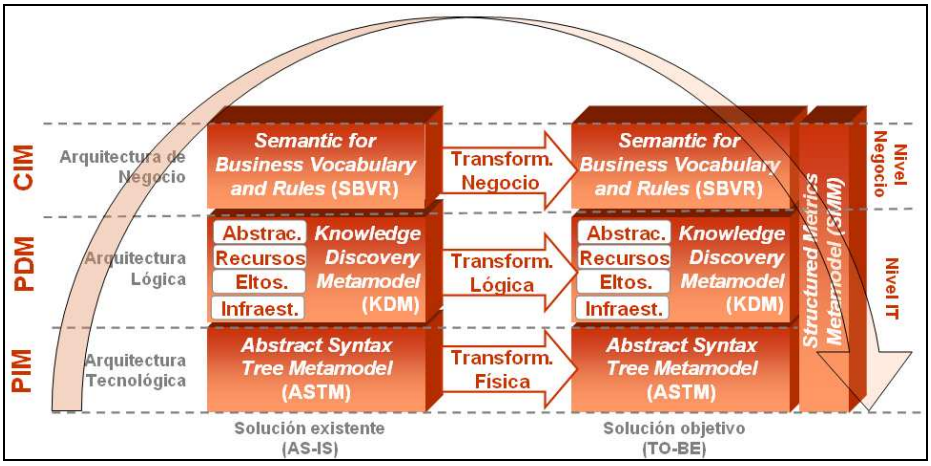


Fig 1 Propuesta de metamodelos y tipos de transformación en el enfoque ADM

La iniciativa ADM contempla la modernización como un conjunto de transformaciones para llegar a una solución objetivo (también denominada TO-BE), partiendo de una solución existente (también llamada AS-IS). Estas transformaciones se pueden aplicar a tres niveles (como puede observarse en la figura 1):

- La transformación más simple, corta y por lo tanto menos compleja, es la correspondiente a una modernización de tipo físico, caracterizada por aportar valor no tanto al negocio, como a la infraestructura empresarial.
- La transformación o modernización lógica correspondería a proyectos de complejidad media, en los que se busca un cambio o una mejora arquitectónica, como puede ser la de transformar un Sistema basado en una Arquitectura Tecnológica de 3-capas a un enfoque SOA.
- Por último, la modernización más compleja, la transformación basada en el Negocio, es aquella que implica a los tres niveles (o al menos los dos superiores), que estaría dirigida por la estrategia de la Empresa, y que busca sobre todo aportar valor al Negocio. Es la más utilizada en los proyectos de Reingeniería.

Para abordar una modernización completa, la mayoría de propuestas (OMG en [2] o Ulrich et al en [7], por ejemplo), recomiendan partir del código fuente para obtener (o lo que es lo mismo, descubrir) un modelo lógico, que correspondería a la Arquitectura Tecnológica del Sistema (Aplicaciones + Datos), y que se puede representar mediante el modelo KDM. El siguiente paso consiste en utilizar el modelado de nivel físico (ASTM) para refinar aquellos aspectos del modelo inicial más cercanos al nivel de programación, sobre todo si la modernización del Sistema implica un cambio de plataforma o lenguaje. A continuación se deberán aplicar las reglas de transformación para obtener un modelo más cercano al Negocio, que será el nexo entre las soluciones existente y objetivo. Por último se deberán aplicar las transformaciones en sentido "descendente" para recorrer el camino de forma inversa en la solución objetivo, partiendo de las nuevas reglas de negocio.

Sin embargo las propuestas de modernización revisadas no ofrecen un enfoque adecuado al caso de estudio que nos ocupa en este trabajo, ya que las que no ofrecen un enfoque teórico, no abordan un ciclo completo o no tienen en cuenta la estrategia de la empresa como objetivo final para alinear negocio y tecnología.

3 Propuesta para la modernización de Sistemas, basada en la Estrategia de Negocio

Por tanto, una vez estudiados los trabajos previos, las recomendaciones del OMG y el enfoque de modernización ADM, este trabajo presenta nuestra propuesta de modernización, basada en el desarrollo orientado a modelos, que se compone de las transformaciones mostradas en la figura 1, y que son descritos a continuación. La figura 2 ilustra los pasos de la propuesta.

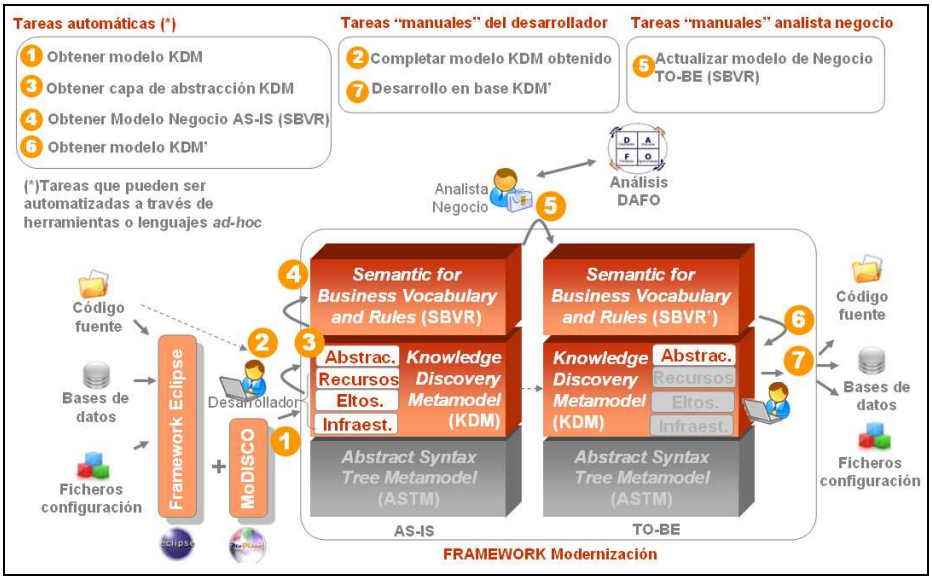


Fig 2 Propuesta de modernización basada en el enfoque ADM

3.1 Obtención del modelo lógico

El primer paso de nuestra propuesta es representar el modelo lógico del sistema actual (AS-IS), utilizando la notación del metamodelo KDM definida en la iniciativa ADM. Para construirlo será conveniente utilizar alguna herramienta dé soporte a la modernización, que reciba como entrada el código fuente, los ficheros de configuración del sistema y los scripts de base de datos.

Puesto que nuestra propuesta está dirigida hacia arquitecturas J2EE, se recomienda la utilización de la herramienta MoDISCO para la extracción semiautomática del modelo KDM.

MoDISCO (Model DISCOvering) es un Framework de la iniciativa Eclipse, descrito en [18], que proporciona una serie de herramientas que facilitan el descubrimiento de los modelos lógicos basados en KDM. A partir del código fuente de un Sistema basado en J2EE, MoDISCO devuelve automáticamente una estructura basada en XMI, que posteriormente puede ser visualizada como un diagrama UML a través del conversor que también ofrece este Framework.

3.2 Refinar el modelo lógico obtenido

A pesar de que MoDISCO puede considerarse una herramienta automática para la obtención del modelo KDM, sólo permite el descubrimiento automático para las capas más cercanas a la tecnología. Como dificultad añadida, este descubrimiento tampoco es completo, ya que los paquetes obtenidos en estas capas no son suficientes para obtener un nivel de abstracción necesario para pivotar hasta el modelo de negocio de forma intuitiva. Por lo tanto, este modelo deberá ser complementado y refinado de forma manual por el desarrollador o el analista, basándose en la inspección visual del código fuente de la aplicación, teniendo en cuenta una serie de normas o reglas básicas, como por ejemplo:

- Un condicional en el código genera un elemento de tipo *ActionElement* (elemento que representa una acción) en el modelo KDM cuyos atributos son: *kind* (operador condicional utilizado) y *value* (expresión derecha a comparar).
- La rama "verdadero"/"falso" de un condicional genera una asociación entre el *ActionElement* en el modelo que lo origina y el *ActionElement* que se ejecuta si se cumple/no se cumple la condición lógica.
- Una instrucción de tipo "mensaje por pantalla" genera un *ActionElement* en el modelo cuyo atributo *value* corresponde al mensaje a mostrar

3.3 Obtener la capa de abstracción del modelo lógico

Mediante la aplicación de reglas de transformación al modelo lógico refinado, se obtendrá de forma automática una representación cercana al negocio del modelo KDM, correspondiente a la capa de abstracción, que permitirá en el siguiente paso obtener las reglas de negocio del sistema existente. Algunas de ellas son:

- Un elemento de tipo *ClassUnit* (clase de diseño) en el modelo lógico genera un elemento *ScenarioUnit* (escenario de comportamiento) en la capa de abstracción.
- Cada *MethodUnit* (método de clase) del modelo lógico genera un *BehaviourUnit* en la capa de abstracción, que herada del *ScenarioUnit* raiz, y un *TermUnit* (término del vocabulario) que hereda del *BehaviourUnit* (modelo de comportamiento) correspondiente
- Cada atributo *value* de un *ActionElement* del modelo lógico genera un *TermUnit* en la capa de abstracción, que hereda de su *BehaviourUnit*.

3.4 Obtener el modelo de negocio de la solución actual

Si bien existe una *task force* para automatizar el proceso de transformación entre el nivel de abstracción de KDM y el modelo SBVR, en el momento de escribir este artículo no se dispone de ningún Framework que permita esta obtención de forma automática. Basándonos sin embargo en la capa de abstracción del modelo KDM, se proponen en el presente trabajo las reglas de transformación necesarias para obtener de forma automática el SBVR, que permitirá derivar las *Business Rules* inherentes al sistema. De esta forma tendremos el nexo de unión a alto nivel entre las soluciones existente y objetivo, además de poder extrapolar el conocimiento del sistema de una forma más inteligible para los expertos de negocio y la Alta Dirección. Algunas de estas reglas son las siguientes:

- Cada *RuleUnit* (rol que juega una clase) de la capa de abstracción se convierte en un elemento *Rule* en el modelo SBVR
- Cada *ActionElement* que implementa un *RuleUnit* genera un *Fact* (hecho a ejecutar) que tendrá como parámetros de tipo *concept* todos los *TermUnit* que implementan un *DataElement* (elemento de base de datos).

Este será por lo tanto, el punto de partida para establecer la estrategia de la modernización basada en las políticas y la visión de la Organización, que se encargará de dirigir el resto del proceso.

3.5 Refinar el modelo de negocio

Mediante técnicas del tipo DAFO [21] se identificarán las debilidades, amenazas, fortalezas y oportunidades a nivel de negocio, para establecer una estrategia que responda a las preguntas: ¿Es necesario cambiar la Tecnología actual? ¿Qué reglas de negocio deben añadirse, cambiarse o eliminarse? ¿Qué enfoque se va a dar a la modernización?, etc. Todo ello estableciendo una priorización y una planificación que permita a la empresa contar con una hoja de ruta clara, y añadir valor al proceso. Este valor puede venir dado como ROI (Retorno de Inversión), satisfacción del cliente/proveedor, posicionamiento en el mercado, etc.

Las nuevas reglas de negocio deberán ser consensuadas entre los analistas del negocio y la dirección de la empresa. Este proceso no es automático y de él dependerá el éxito del proyecto. Además, en un enfoque SOC, estas reglas serán las mejores candidatas para publicar los correspondientes servicios de valor añadido para la entidad.

3.6 Obtener el modelo lógico de la solución objetivo

Una vez definidas las nuevas reglas de negocio y determinada la estrategia a seguir, el siguiente paso será aplicar de forma automática las reglas de transformación que permitan obtener la capa de abstracción del modelo KDM de la solución objetivo, partiendo del modelo SBVR refinado en el paso anterior. Esta fase podrá ser realizada de forma iterativa, según la estrategia establecida.

3.7 Implementar el modelo lógico de la solución objetivo

Partiendo de la capa de abstracción del modelo lógico, el último paso consiste en implementar el código fuente de la solución objetivo. Este paso puede ser realizado de forma manual por el desarrollador, partiendo del modelo lógico obtenido en el paso anterior, y utilizando normas y reglas básicas como las descritas en el apartado 3.2.

4 Caso de Estudio: Transmisión de dominios .es

Para comprobar la viabilidad de la propuesta planteada en este trabajo, se ha decidido aplicarla a un caso práctico: el sistema para transmisión de nombres de dominio .es. Red.es, como autoridad de asignación delegada por IANA e ICANN [14] para la gestión de los nombres de dominio en España, cuenta con un sistema informático, que inicialmente fue concebido para la gestión de unos pocos .es, pero que en la actualidad cuenta con más de 1,2 millones de nombres de dominio.

Para adaptarse a esta situación, el sistema ha ido evolucionando bajo demanda, mediante la aplicación de sucesivos parches, lo que ha ido degradando la eficiencia del mismo, incurriendo en unos elevados costes de mantenimiento.

Las necesidades operativas han hecho a Red.es plantearse la modernización del sistema actual, que permita a la Entidad adelantarse en la implantación de la eAdministración que el Gobierno de España ha marcado para el año 2015 [16].

Tras distintos análisis, se ha decidido que el enfoque SOC basado en modelos que propone ADM es el más adecuado para este proyecto. Puesto que el sistema actual es demasiado complejo, centraremos el objeto del caso de estudio en un subconjunto del mismo, más concretamente en la funcionalidad para el cambio de titularidad de un dominio .es (o transmisión de dominio). Veamos a continuación los pasos de los que consta el proceso de modernización aplicado al caso de estudio.

4.1 Obtención del modelo lógico

El primer paso del proceso de modernización es obtener el modelo KDM automáticamente a partir del código fuente del sistema actual. Para ello se ha utilizado la herramienta MoDISCO, obteniendo como resultado un modelo lógico preliminar como el de la figura 3, en el que se muestra una captura de pantalla correspondiente a la representación visual del KDM obtenido con esta herramienta.

4.2 Refinar el modelo lógico obtenido

Tomando como base el modelo KDM obtenido, se ha realizado el refinamiento del mismo de forma manual (por parte del desarrollador) incorporando, tras realizar una inspección del código fuente, aquellos elementos no contemplados por MoDISCO Tal es el caso de las asociaciones de tipo *TrueFlow*, *FalseFlow*, *read* y *write* entre los elementos de tipo *ActionElement*, que representan las principales acciones que ejecuta un programa, o de los atributos *kind* y *value* de este tipo de elementos.

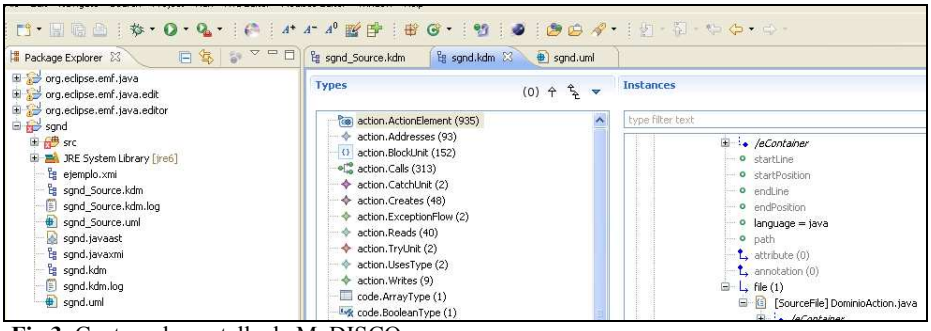


Fig 3. Captura de pantalla de MoDISCO

Mediante la revisión de las ramas condicionales del código, y tomando como base las reglas propuestas, el desarrollador modifica el modelo KDM (manualmente) para incorporar estos elementos, con objeto de conseguir un modelo refinado como el mostrado en la figura 4.

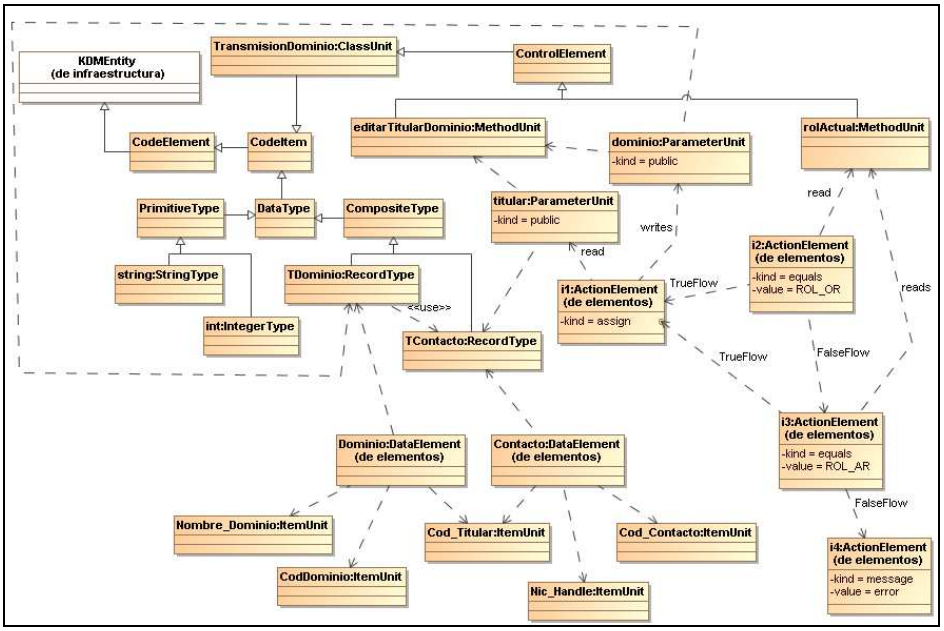


Fig 4 Modelo KDM refinado por el desarrollador

4.3 Obtener la capa de abstracción del modelo lógico

Una vez obtenido el modelo KDM refinado, el siguiente paso es generar la capa de abstracción del modelo, mediante la aplicación de las reglas de transformación propuestas. Así, para el caso de estudio, el elemento *editarTitularDominio* (de tipo

MethodUnit) genera un elemento *BehaviourUnit* que hereda de *TransmisionDominio*, (que a su vez es un *ScenarioUnit* derivado del *ClassUnit* del modelo KDM).

A partir de ahí, se generan los elementos de tipo *TermUnit ROL_AR*, *ROL_OR* (correspondientes a atributos de tipo value dentro de los *ActionElement* obtenidos en el modelo KDM refinado), dominio y contacto (derivados de los elementos de tipo *ParameterUnit* dentro de este modelo). Se generará también el elemento *PuedeEditarTitularDominio*, de tipo *FactUnit*, que representa el hecho principal del escenario, y que deriva del *MethodUnit* del KDM refinado que se está construyendo, y que centraliza todas las asociaciones de tipo *TrueFlow*, *FalseFlow*, *read* y *write* con los elementos de tipo *ActionElement* y/o *MethodUnit* del modelo KDM refinado. Todo ello derivará del elemento raíz *BehaviourUnit*. La figura 5 muestra el modelo final obtenido.

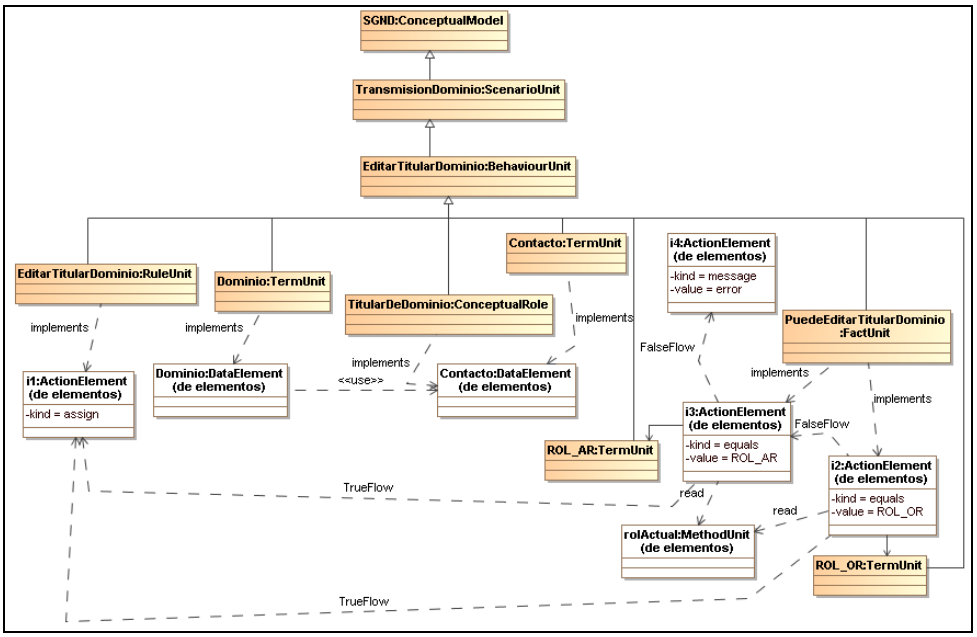


Fig 5 Capa de abstracción del modelo KDM

4.4 Obtener el modelo de negocio de la solución actual

Partiendo de la capa de abstracción, el objetivo de esta fase consiste en aplicar las reglas de transformación que permitan obtener el modelo SBVR del caso de estudio. Este modelo es relativamente fácil de conseguir, ya que los elementos de tipo *fact* marcan las condiciones de la regla de negocio. La figura 6 muestra el modelo obtenido para nuestro caso de estudio.

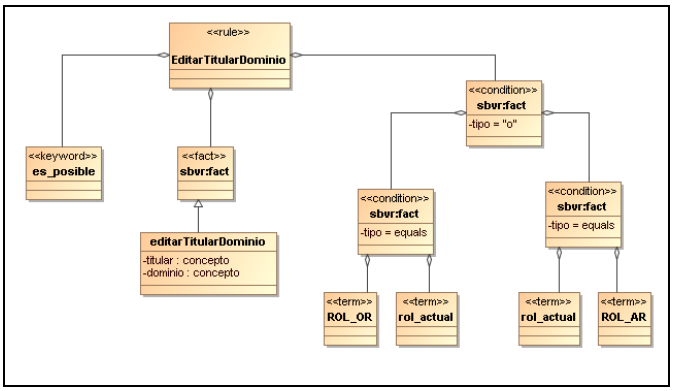


Fig 6 Modelo SBVR del sistema origen

4.5 Refinar el modelo de negocio

Tras realizar un análisis DAFO, encargado por la Alta Dirección y llevado a cabo por los analistas de negocio, se ha realizado un estudio de los sistemas de distintos Registros europeos. Como principal debilidad (que a su vez se constituye como una oportunidad), se ha detectado la necesidad de adaptar el sistema a las nuevas leyes y normas de interoperabilidad [15] y de acceso electrónico de los Ciudadanos a los Servicios Públicos [16].

Corresponde a los analistas de negocio modelar las actuales reglas para contemplar este nuevo aspecto. De esta forma, se ha actualizado el modelo SBVR, de forma manual, para incorporar una nueva rama que permita al usuario solicitar la transmisión de un dominio siempre que se cumpla la condición de que dicho usuario sea el titular del mismo, y que éste se haya autenticado con certificado digital (nuevos elementos *fact* en el modelo). La figura 7 muestra el modelo SBVR correspondiente a la solución objetivo.

4.6 Obtener e implementar el modelo lógico de la solución objetivo

A partir del modelo de negocio de la solución objetivo, el desarrollador deberá tener en cuenta las normas y reglas básicas (como las descritas en el apartado 4.2) para obtener el modelo lógico de la solución objetivo. En el caso de estudio se ha realizado de forma manual, pero sería susceptible de ser automatizado en trabajos futuros.

4.7 Implementar el modelo lógico de la solución objetivo

El último paso consistirá en codificar la solución objetivo, partiendo del modelo lógico, y aplicando las reglas de transformación correspondientes (que en nuestro caso de estudio se han aplicado de forma manual, pero que serían susceptibles de ser automatizadas. Además, estas reglas serán candidatas a ser publicadas como nuevos servicios de valor para la organización. De esta forma, la nueva regla de negocio en

nuestro caso de estudio, se ha expuesto como un *wrapper*, para poder ofrecerlo a los ciudadanos sin demasiado coste de implementación en el menor tiempo posible.

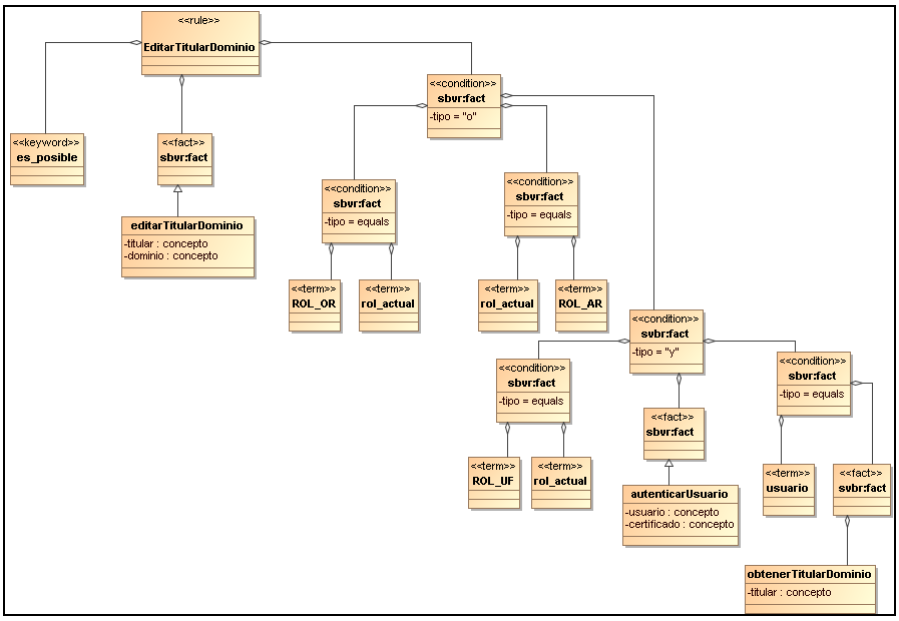


Fig 7 Modelo SBVR del sistema objetivo

5 Conclusiones y trabajos futuros

La propuesta de modernización planteada en este trabajo, basada en el desarrollo dirigido por modelos, toma como punto clave la definición de una estrategia que se encargue de alinear negocio y tecnología. En base a la misma, se ha tomado un caso de estudio real, disponible en http://www.dominios.es.

Las principales ventajas que este proceso ha añadido al caso de estudio, han sido la disminución de los costes y los tiempos de implementación, (y a medio plazo los de mantenimiento), la alineación con la estrategia de la empresa, y el aumento de la satisfacción del usuario, al permitir la transmisión de sus dominios "sin papel", adelantándose 4 años a los hitos de eAdministración marcados por el Gobierno de España. Queda pendiente, sin embargo, la incorporación al marco de trabajo de los indicadores necesarios, como parte del modelo SMM, como parte de un proceso de mejora continua.

Otras líneas de continuación pasarían por intentar automatizar los pasos que actualmente se realizan de forma manual (excepto el del refinamiento de las reglas de negocio que, por su complejidad no puede ser automatizado), construir las herramientas necesarias para aplicar las reglas de transformación entre modelos de forma automática, independizar el marco de trabajo de la plataforma, incorporando el modelo ASTM para la representación del código de la aplicación, y la utilización del lenguaje natural para la representación del modelo SBVR, así como diseñar las

herramientas necesarias para contemplar gramáticas que permitan la transformación automática entre ambas representaciones.

Destacar que este trabajo se ha desarrollado en el marco de los proyectos MODEL CAOS (TIN2008-03582) y Agreement Technologies (CONSOLIDER CSD2007-0022) financiado por el Mº de Ciencia y Tecnología de España.

7 Referencias

1. Especificación MDA del OMG, disponible en http://www.omg.org/mda/specs.htm
2. Especificación ADM del OMG disponible en http://adm.omg.org/
3. Especificación KDM del OMG disponible en www.omg.org/technology/kdm/index.htm
4. Chung, S., Byung, J. Service-Oriented Software Reengineering: SoSR. IEEE Proceedings of th 40th Hawaii International Conference on Systems Science (2007)
5. Pu, J, Yang, H., Xu, B., Xu, L., Cheng-Chung, W. Combining MDE and UML to Reverse Engineer Web Based Legacy Systems. Annual IEEE International Computer Software and Applications Conference (2008)
6. J.M. Favre, "Foundations of Model (driven) (Reverse) Engineering - Episode I: Story of the Fidus Papyrus and the Solarus", post-proceedings of Dagsthul Seminar on Model Driven Approaches for Language Engineering, May 2004
7. Information Systems Transformation: Architecture-Driven Modernization Case Studies By: William M. Ulrich; Philip H. Newcomb Publisher: Morgan Kaufmann
8. R. Pérez-Castillo et al., Business process archeology using MARBLE, Inform. Softw. Technol. Mayo 2011
9. Cánovas Izquierdo, J., García Molina, J. Extracción de Modelos en una Modernización basada en ADM. Actas de los talleres de las Jornadas de Ingeniería del Software y BBDD. Vol. 3, No. 2 (2009)
10. Especificación SBVR del OMG disponible en http://www.omg.org/spec/SBVR/1.0/
11. Baxter, I.; "A Standards- Based Approach to Extracting Business Rules" Architecture Driven Modernization Workshop, Alexandria. Octubre 2005
12. Van den heuvel, W. "Matching and Adaptation: Core Techniques for MDA-(ADM)-driven Integration of new Business Applications with Wrapped Legacy Systems". Model Driven Evolution of Legacy systems. Septiembre 2004
13. Rajala, M. "Minimizing the Risks of Architecture Driven Modernization Projects" Architecture Driven Modernization Workshop, Chicago, Marzo 2004.
14. BOE número 99 de 25/4/1998. Ley 11/1998, de 24 de abril, General de Telecomunicaciones (Disposición Adicional Sexta). Páginas 13909 a 13940
15. BOE número 150 de 23/06/2007. Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos. Páginas 27150 a 27166
16. Real Decreto 4/2010, de 8 de enero, por el que se regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica
17. Especificación MOF del OMG disponible en http://www.omg.org/mof/
18. MoDISCO user guide. Disponible en http://help.eclipse.org/helios/index.jsp?nav=/38
19. Especificación ASTM del OMG disponible en http://www.omg.org/spec/ASTM/1.0/
20. Especificación SMM del OMG disponible en http://www.omg.org/spec/SMM/
21. Nakajima, S.; "Semi-automated diagnosis of FODA feature diagram". Proceedings of the 2010 ACM Symposium on Applied Computing. Marzo 2010
22. Sánchez, O., Sánchez J., García J.; "Model Driven Reverse Engineering of legacy graphical user interfaces". Proceedings of the IEEE/ACM international conference on Automated software engineering. Septiembre 2010

Ingeniería de requisitos orientada a servicios: características, retos y un marco metodológico[1]

Marcela Ruiz[1], David Ameller[2], Sergio España[1], Pere Botella[2], Xavier Franch[2], Oscar Pastor[1]

[1] Centro de Investigación en Métodos de Producción de Software (ProS)
Universitat Politècnica de València
{lruiz, sergio.espana, opastor}@pros.upv.es
[2] Grupo de investigación en Ingeniería del Software para los Sistemas de Información (GESSI)
Universitat Politècnica de Catalunya
{dameller, botella, franch}@essi.upc.edu

Abstract. La ciencia de los servicios es, más que una nueva disciplina, un enfoque interdisciplinar para el estudio, diseño, e implementación de sistemas orientado a servicios, que actúa como paraguas que cubre todos los aspectos de computación utilizados (es decir, especificación y diseño orientado a servicios, arquitecturas orientada a servicios, servicios web, etc.), siendo actualmente una de las áreas de investigación más activas en el ámbito de la informática. La provisión de servicios y la innovación de los mismos están basadas sobre todo en las tecnologías de información. Este artículo presenta un análisis de las características esenciales e inherentes a la orientación a servicios. En base a este análisis, se identifican carencias en los métodos y herramientas de ingeniería de requisitos actuales que dificultan su aplicación al paradigma orientado a servicios. Estas carencias contribuyen a trazar un plan de trabajo con los retos que la Ingeniería de Requisitos Orientada a Servicios (IROS) deberá enfrentar los próximos años. Por último, se ofrece un marco de IROS con el potencial para dar solución a los retos encontrados. Este marco enfatiza, además, la importancia de la reutilización de métodos y conocimiento existentes mediante una estrategia de macromodelado, así como la aplicabilidad de las aproximaciones de desarrollo dirigido por modelos.

Keywords: Ingeniería de requisitos, ciencia de los servicios, desarrollo de software dirigido por modelos, ingeniería de requisitos orientada a servicios.

1 Motivación

La *Ciencia de los Servicios* (*Service Science*) es un término de reciente aparición en el ámbito académico que propone un enfoque interdisciplinar para el estudio, diseño, e

[1] Investigación soportada por el MICINN y la GVA bajo los proyectos Pros-Req (subproyectos TIN2010-19130-C02-01 y TIN2010-19130-C02-02) y PROMETEO/2009/015, beca FPI programa Santiago Grisolía de la GVA Orden 62/2011 y cofinanciado con FEDER.

implementación de sistemas orientado a servicios. Otros términos generalmente usados como sinónimos son *Service Science, Management and Engineering* (SSME) o también SSMED (donde la 'D' denota *Design*). Empezó como un "call-to-action" lanzado por compañías como IBM, HP, Oracle y otras hacia el mundo académico, para revisar qué disciplinas ya desarrolladas eran aplicables a los sistemas de servicios, y en su caso, desarrollar nuevas disciplinas. Tras pocos años transcurridos existen ya entidades de referencia como el SRII (Service Research and Innovation Institute, http://www.thesrii.org) que centran su interés en el concepto de "*IT-enabled services*" es decir, aquellos servicios en los que las tecnologías de la información desempeñan un papel relevante. El sector de los servicios representa el mayor porcentaje de la economía en los países desarrollados, y es innegable que una gran parte de la innovación en servicios se basa en la tecnología informática. Y ésta, por su parte, ha contribuido con aportaciones que configuran la base tecnológica para la implementación de servicios: arquitecturas orientadas a servicios (SOA), servicios web o computación en la nube (*cloud computing*) son conceptos que obligan a revisar los paradigmas de desarrollo de software. Esta revisión necesaria es la respuesta al "*call-to-action*" arriba mencionado. Del mismo modo que en el ámbito de la gestión (*management*) se está evolucionando de una lógica basada en el producto a una basada en el servicio [1] (de forma que se están reformulando conceptos como marketing, innovación, sostenibilidad, ciclo-de-vida, etc. de producto a servicio), en el ámbito de la informática es también necesario revisar todo lo existente en ingeniería del software para el desarrollo de sistemas de información clásicos, y adaptarlo para el desarrollo de sistemas de servicios.

Este artículo se enfoca en la Ingeniería de Requisitos (IR), el paso inicial en un proyecto de sistema informático, es decir, la etapa del ciclo de vida que tiene como objetivo el desarrollo de una especificación completa, consistente y no ambigua del sistema. La motivación del presente artículo surge de la necesidad de involucrar métodos y técnicas actuales de la IR a sistemas orientados a servicios; lo que nos lleva a plantearnos la posibilidad de establecer un marco específico en el cual todas las propuestas de IR, interactúen y se encuentre adaptadas para el desarrollo de sistemas de información orientados a servicios.

2 La naturaleza de los servicios

La ciencia de los servicios es muy reciente (los primeros trabajos datan de 2004 y 2006), aunque se fundamente en disciplinas pre-existentes. Ello conlleva a que la mayor parte de sus conceptos no estén todavía completamente asentados. Empezando por el concepto de "servicio", no solo encontramos muchas definiciones, sino que podemos afirmar que se trata de un concepto polisémico. Una definición bien aceptada en ciencia de los servicios es la de Vargo y Lusch [1] en la que un servicio es "la aplicación de competencias (conocimientos y habilidades) a través de acciones, procesos y actuaciones en beneficio de otra entidad o de la propia entidad". Esta definición se contrapone a las definiciones clásicas en ciencias de la gestión, que siempre ponen por delante dos aspectos: la consumación simultánea entre productor y

consumidor, y la intangibilidad [2]. En esta visión clásica se suelen describir cuatro características de los servicios: intangibilidad, inseparabilidad, heterogeneidad y la condición de perecedero. También es complejo tratar de encontrar una definición de "sistema de servicios". Alter [2] presenta un sistema de servicios como un instrumento común a las disciplinas del marketing, las operaciones y la informática para comprender, analizar y diseñar servicios, y lo vincula a tres marcos. Una referencia básica es [3] en donde un sistema de servicios es "una configuración de personas, tecnologías y otros recursos que interactúan con otros sistemas de servicios para la creación de valor mutuo". Es interesante remarcar que esta definición, en donde aparecen personas y tecnología, nos recuerda la visión clásica de un sistema de información, en donde también personas y tecnología interactúan para recibir, almacenar, procesar y producir información. Otros autores también señalan este paralelismo entre sistemas de servicios y sistemas de información [4]. Es éste un aspecto de gran relevancia para este trabajo ya que, partiendo de esta aparente similitud de los sistemas de servicios con los sistemas de información, podemos empezar analizando la IR pensada para los sistemas de información y ver en qué aspectos hay que adaptarla para tratar con sistemas de servicios. En definitiva se trata de recoger el reto del "*call-to-action*" de la ciencia de los servicios, y tener en cuenta que éstos tienen características propias, nuevas y diferentes de las que estamos acostumbrados a tratar.

3 Identificación de retos y definición del plan de trabajo

La comunidad de IR tiene por delante una serie de retos que debe afrontar para proporcionar un soporte adecuado al paradigma de la orientación a servicios. Como se indica en [5], "*Service-Oriented Requirement Engineering differs from traditional RE as it assumes that the concerned application will be developed in an SOA framework running in an SOA infrastructure*". Nuestro objetivo es diseñar un marco específico para la IROS que aúne los conceptos generales de la IR con las especificaciones propias de la orientación a servicios. A continuación detallamos ambas dimensiones.

Por lo que se refiere a conceptos generales, las aproximaciones metodológicas de IR se pueden clasificar bajo diferentes perspectivas, que son a la vez necesarias (cada una proporciona un valor en particular) y complementarias (todas ellas en su conjunto abarcan la naturaleza poliédrica de las necesidades expresadas en los requisitos).

En lo que a las particularidades de la orientación a servicios se refiere, destacamos la necesidad de considerar el uso de servicios ya existentes desde la fase inicial de educción de requisitos, por este motivo los requisitos de partida de los usuarios deben ser más flexibles.

Como consecuencia de este análisis, identificamos dos retos principales:

- Identificar las perspectivas dominantes en la IROS y proporcionar un marco metodológico y formal adecuado para su integración.
- Integrar la oferta de servicios en los procesos de la IROS.

La complejidad de estos retos en el contexto de la IROS es elevada. Cabe destacar que la mayoría de sistemas basados en servicios se caracterizan por la existencia de

un elevado número de partes interesadas (*stakeholders*), que por lo general tienen requisitos diferentes, potencialmente conflictivos y en ocasiones efímeros, cuya gestión es por lo tanto especialmente compleja. En la sección siguiente presentamos una propuesta que se dirige a los retos identificados.

4 Solución propuesta: un marco de IROS

Considerando los retos encontrados y expuestos en la Sección 3, se propone un marco para la IROS concebido con varios objetivos en mente: (i) facilitar la captura de requisitos sobre sistemas orientados a servicios, (ii) reutilizar metodologías y conocimientos existentes, (iii) diferenciar la vista funcional y no funcional del sistema de información. La Fig. 1 presenta una panorámica general del marco propuesto.

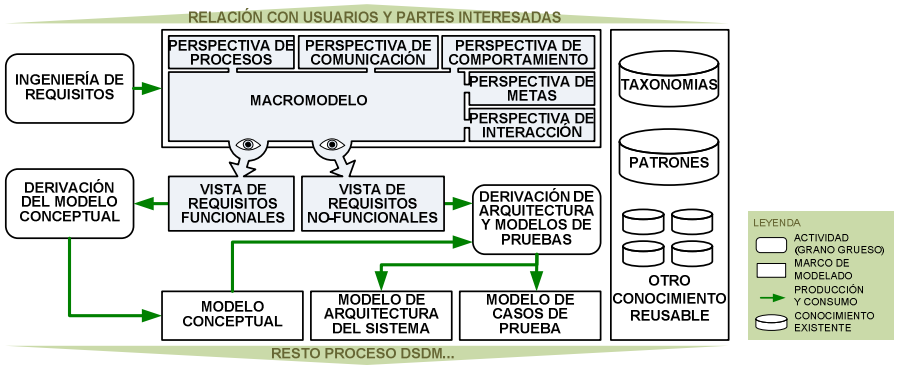


Fig. 1. Marco de Ingeniería de Requisitos Orientado a Servicios (IROS).

Los analistas obtienen, mediante técnicas de captura de requisitos, información sobre el sistema y las necesidades de los usuarios y otras partes interesadas. El marco de IROS propuesto, provee a los analistas diversas perspectivas para analizar y especificar los requisitos: se planea ofrecer perspectivas de procesos (¿cuál es el flujo de los servicios?, ¿qué actividades se llevan a cabo?), de comunicación (¿qué comunicaciones se producen entre la organización y los usuarios?), de comportamiento (¿qué tan rápido, fiable y seguro, etc., es el sistema en producción?), de metas (¿por qué se toman las decisiones?, ¿de dónde vienen?, ¿qué alternativas se han considerado?) y de interacción (¿cuáles son las interacciones entre el sistema y sus usuarios?, ¿cuándo se llevan a cabo?). La adecuada integración de las diferentes perspectivas se facilita mediante una estrategia de macromodelado [6].

El macromodelo también permite diferenciar dos vistas de los requisitos modelados: la vista de los requisitos funcionales y la vista de los requisitos no-funcionales. La importancia de distinguirlos, además de que responden a distintas aproximaciones metodológicas, reside en su conexión con diferentes actividades del proceso de desarrollo dirigido por modelos (DSDM). Los requisitos funcionales permiten derivar un modelo conceptual que recoge los aspectos estructurales, dinámicos y (en caso necesario) de interfaz. A partir del modelo conceptual y de los

requisitos no-funcionales, es posible derivar un modelo de arquitectura (p.ej. orientada a servicios, SOA), al tiempo que se derivan modelos de casos de prueba que permitirán llevar a cabo tareas de *testing* de forma semi-automática con el fin de garantizar la calidad del sistema software final mediante técnicas de prueba de software basado en requisitos. El marco propuesto, dispone de un repositorio de conocimiento que puede ser reutilizado durante las tres actividades de DSDM mencionadas [7]. Este conocimiento puede estar representado mediante ontologías para sistemas software en contextos DSDM [8]; propuestas existentes como SWEBOK, ofrecen un soporte ontológico muy general y abstracto para entornos DSMD y en especial ambientes SOA [9]. Por otro lado, los patrones de requisitos, de diseño y arquitectura son otra forma de representar el conocimiento existente, la aproximación [10] puede ser adaptada a esta propuesta. Existe otro conocimiento reusable que puede no estar representado como ontologías o patrones (p.ej. documentación legislativa, manuales de uso y metodologías de procesos), es importante realizar un análisis sobre si es integrable con técnicas de captura de requisitos y metodologías de DSDM, para así, garantizar el éxito de la reutilización de conocimiento.

5 Discusión y conclusiones

El sector de los servicios representa el mayor porcentaje de la economía en los países desarrollados, y es innegable que una gran parte de la innovación en servicios se basa en la tecnología informática. Y ésta, por su parte, ha contribuido con aportaciones que configuran la base tecnológica para la implementación de servicios: arquitecturas SOA, servicios web o computación en la nube (*cloud computing*) son conceptos que obligan a revisar los paradigmas de desarrollo de software.

El marco de IROS presentado en la sección 4, abre diversas ventanas de discusión en torno a sus diferentes aspectos y características principales; por ejemplo, la necesidad de proporcionar un enfoque de macromodelado para unificar las vistas del sistema, la participación del paradigma de DSDM y los retos relativos a la provisión de mecanismos para la derivación de los modelos conceptuales y su refinamiento; adicionalmente, el marco de IROS debe incorporar técnicas para la evaluación de los modelos conceptuales y el *testing* de los sistemas orientados a servicios generados, con el fin de asegurar su calidad. Estos temas son desarrollados con mayor detalle a continuación.

El enfoque de macromodelado actúa como soporte para suplir la necesidad de que las perspectivas allí involucradas estén presentes como piezas complementarias. Sin embargo, esto introduce dos retos. En primer lugar las perspectivas deben ser analizadas y adaptadas según el desarrollo del sistema software orientado a servicios que se va a implementar [5]. En segundo lugar, es necesario sincronizar todas las perspectivas y articularlas alrededor de un marco apropiado para su interacción. Por otro lado, en el paradigma del DSDM, los modelos conceptuales corresponden a un nivel de abstracción diferente al de los modelos de requisitos. Frecuentemente, el analista necesita añadir información adicional para poder obtener una especificación

más completa del sistema a desarrollar. Por lo tanto, se hace necesaria la existencia de mecanismos apropiados para realizar el refinamiento necesario, especialmente en un contexto orientado a servicios.

La evaluación de los modelos conceptuales obtenidos del proceso de transformación, se debe concentrar en la completitud y corrección de dichos modelos; se hace necesario contar con mecanismos de evaluación y *testing*, que se enfoquen en la monitorización de los requisitos y servicios del sistema, cuya naturaleza cambiante hace necesaria la evolución y adaptación de las herramientas y modelos de prueba, con el fin de asumir la evolución intrínseca de los sistemas orientados a servicios.

Cabe notar que, en este punto de la investigación, la propuesta no pretende abordar la implementación manual o la generación automática del software final. Se ha optado por llegar hasta el nivel de la arquitectura, en el cual se toman las decisiones más importantes con respecto a los requisitos no funcionales. Una vez se ha obtenido el modelo de arquitectura, las tareas de generación de código (si bien, siguen siendo complejas) podrían ser automatizadas.

Referencias

1. S. L. Vargo and R. F. Lusch. Evolving to a New Dominant Logic for Marketing. *Journal of Marketing,* vol. 68(1), 2004, pp. 1-17.
2. S. Alter. Service system fundamentals: work system, value chain, and life cycle. *IBM Systems Journal,* vol. 47(1), January 2008, pp. 71-85.
3. J. Spohrer, S. L. Vargo, N. Caswell, and P. P. Maglio "The Service System Is the Basic Abstraction of Service Science," in *Hawaii International Conference on System Sciences HICSS '08*, 2008.
4. J. Gou, X. Li, X. Li, and P. Zhao, "Discipline Comparison of SSME with IS and its Education Implications," in *IEEE Congress on Services - Part I SERVICES '08*, 2008, pp. 57-61.
5. W. T. Tsai, Z. Jin, P. Wang, and B. Wu, "Requirement Engineering in Service-Oriented System Engineering." In: IEEE International Conference on eBusiness Engineering ICEBE07, Hong Kong, China, 2007.
6. R. Salay, J. Mylopoulos, and S. Easterbrook, "Using Macromodels to Manage Collections of Related Models." In: 21st International Conference on Advanced Information Systems (CAiSE), 2009.
7. P. Mohagheghi and R. Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering,* vol. 12(5), 2007.
8. D. Ameller and X. Franch, "Definición de una Ontología para el Proceso de DSDM considerando Requisitos No-Funcionales." In: Desarrollo de Software Dirigido por Modelos (DSDM), 2008.
9. R. C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc, and A. Jansen, "Architectural Knowledge: Getting to the Core " In: 3rd international conference on Software architectures, components, and applications QoSA, 2007.
10. S. Renault, Ó. Méndez-Bonilla, X. Franch, and C. Quer. A Pattern-based Method for building Requirements Documents in Call-for-tender Processes. *International journal of computer science & applications (IJCSA),* vol. 6(5), 2009, pp. 175-202.

Desarrollo de servicios con SoaML desde procesos de negocio en BPMN: metodología y automatización

Andrea Delgado[1], Ignacio García-Rodríguez de Guzmán[2], Francisco Ruiz[2]

[1] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, 1300 Montevideo, Uruguay
adelgado@fing.edu.uy

[2] Grupo Alarcos, Depto. de Tecnologías y Sistemas de Información, Universidad de Castilla – La Mancha, Paseo de la Universidad 4, 13071,Ciudad Real, España
{ignacio.grodriguez, francisco.ruizg}@uclm.es

Resumen. La sinergia entre los paradigmas de Gestión de Procesos de Negocio (Business Process Management, BPM) y Computación Orientada a Servicios (Service Oriented Computing, SOC) establecida en los últimos años, aporta al cierre de la brecha negocio-sistemas mediante la realización de procesos de negocio (PN) como servicios. Si bien la implementación y ejecución de servicios es un área que en los últimos años ha madurado considerablemente, el diseño y modelado de servicios aún está en definición. El modelado de servicios es fundamental entre otras cosas, para la automatización de distintas etapas del desarrollo en base al paradigma de Desarrollo Dirigido por Modelos (Model Driven Development, MDD). El estándar SoaML es un paso para el avance en dicha área, definiendo un metamodelo y un perfil UML para el modelado de servicios, con estereotipos específicos. En este artículo presentamos la extensión de nuestra metodología BPSOM para desarrollo de servicios desde PN, integrando dos aspectos principales: el nuevo estándar SoaML para diseño y modelado de servicios, y transformaciones QVT para generación automática de servicios en SoaML desde modelos de PN en BPMN.

Palabras clave: Business Process Management (BPM), Service Oriented Computing (SOC), Model Driven Development (MDD), metodologías de diseño y desarrollo de servicios, generación automática, QVT, BPMN, SoaML.

1 Introducción

La realización de procesos de negocio (PN) con servicios se ha convertido en una de las formas preferidas para implementar el software para informatizar los procesos de negocio en las organizaciones. La aplicación conjunta de los paradigmas de Gestión de Procesos de Negocio (Business Process Management, BPM) [1][2][3] y de Orientación a Servicios (Service Oriented Computing, SOC) [4] provee la base conceptual y técnica, para acercar la visión de las áreas del negocio y software en pos del objetivo común que es poder realizar el negocio que define a la organización de la forma más adecuada, permitiendo la introducción de cambios tanto en los PN como

en el software asociado, minimizando el impacto de dichos cambios y permitiendo su incorporación con la mayor agilidad posible. La Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA) [5][6] es un estilo de arquitectura que soporta el paradigma de desarrollo SOC, usualmente implementado con Servicios Web (Web Services, WS). Si bien la implementación y ejecución de servicios es un área que en los últimos años ha madurado considerablemente, el diseño de modelos de servicios aún está en definición. El modelado de servicios es fundamental entre otras cosas, para la automatización de distintas etapas del desarrollo de software utilizando el paradigma de Desarrollo Dirigido por Modelos (Model Driven Development, MDD) [7] que agrega a dicha visión el enfoque centrado en modelos, tanto de PN como de servicios, para relacionar y transformar elementos de un modelo origen con elementos de otro modelo destino, mediante la definición de correspondencias entre los mismos. El enfoque de Arquitectura Dirigida por Modelos (Model Driven Architecture, MDA) [8] es la propuesta de OMG para MDD. El estándar de Modelado de Arquitecturas Orientadas a Servicios (Service Oriented Architecture Modeling Language, SoaML) [9] de OMG es un paso para el avance del modelado de servicios, que define un metamodelo y un perfil UML, extendiendo la notación UML.

La metodología de desarrollo orientado a servicios desde PN (Business Process Service Oriented Methodology, BPSOM) que guía el desarrollo de servicios desde PN, integra la dimensión metodológica del marco MINERVA [10][11] definido para soportar la mejora continua de PN realizados por servicios y basada en modelos, integrando los paradigmas mencionados de BPM, SOC y MDD. MINERVA comprende tres dimensiones en las que se integran conceptos [12], metodologías [13] [14], y herramientas [15], para soportar las definiciones y el trabajo en cada fase definida. En este artículo se presenta la extensión de las definiciones de BPSOM [13], integrando dos aspectos principales para la realización de PN con servicios: el uso del estándar SoaML para modelado de servicios, y la definición de transformaciones con el lenguaje Query/View/Transformations (QVT) [16] entre modelos de PN especificados en la notación de Modelado de Procesos de Negocio (Business Process Modeling Notation, BPMN)[17] y modelos de servicios en SoaML. Esto permite generar automáticamente los servicios necesarios desde los modelos de PN en base a las relaciones identificadas entre elementos de ambos metamodelos.

El resto del artículo está organizado como sigue: en la sección 2 se presenta el modelado de PN y servicios con los estándares BPMN y SoaML asociados mediante el ejemplo que se introduce, y el enfoque que tomamos para su generación automática; en la sección 3 se describe brevemente la metodología BPSOM detallando la utilización del estándar SoaML en cada actividad definida, los diagramas a realizar, los roles responsables, los estereotipos a utilizar, siguiendo el ejemplo introducido en la sección anterior; en la sección 4 se describen los trabajos relacionados y finalmente en la sección 5 se presentan conclusiones y trabajo futuro.

2 Modelado de PN y servicios

Hemos detectado siete principios básicos a tener en cuenta al realizar la integración de los paradigmas SOC y MDD a los procesos de negocio [18]: el modelado de PN y de servicios, transformación de modelos, enfoque metodológico, uso de patrones

(procesos y diseño), procesos colaborativos y herramientas de soporte, que fueron tenidos en cuenta en la definición de MINERVA. Para el modelado de PN y servicios se utiliza una amplia variedad de notaciones existentes, siendo los principales BPMN para PN y UML para servicios. Aunque SoaML es un estándar de reciente definición, es esperable que sea rápidamente incorporado por la comunidad de software para el modelado de servicios, ya que extiende UML que es ampliamente utilizado.

2.1 Elementos clave en BPMN2

La notación para Modelado de Procesos de Negocio (BPMN) [17] de OMG, es una notación estándar para modelar visualmente flujos de procesos que tiene como objetivo proveer notación común para analistas del negocio que crean los flujos iniciales de los procesos y desarrolladores de software responsables de la tecnología e implementación de los procesos. La versión actual (BPMN2) introduce varios cambios a las anteriores, organizando el modelado de PN en base a cuatro construcciones principales: *Process*, *Collaboration* y *Choreography* que integran el núcleo de elementos definidos, y *Conversation* que es un uso particular y una descripción informal de una *Collaboration*. De esta forma es posible especificar una orquestación (*Process*) que define un proceso en los límites de una organización que lo controla, una colaboración (*Collaboration*) que permite especificar la interacción entre procesos de dos o más organizaciones que interactúan en base a mensajes y flujo de mensajes, donde ninguna tiene el control absoluto de la interacción; una coreografía (*Choreography*) que describe en detalle la interacción entre procesos de una colaboración con foco en los mensajes intercambiados, y una conversación que muestra una visión global de una colaboración. Estos elementos se conectan al elemento de definiciones (*Definitions*) que es el elemento raíz en la jerarquía. Se modelan explícitamente los servicios definidos integrando varios elementos como interfaces (*Interfaces),* operaciones *(Operations), y mensajes (Messages)* de entrada y salida y sus relaciones. El resto de elementos de las versiones anteriores de BPMN se mantienen: objetos de flujo, de conexión, swimlanes y artefactos.

En la Fig. 1 se presenta el modelo BPMN del proceso "Admisión y Registro de Paciente para Cirugía Mayor Ambulatoria (CMA)", uno de los PN que estamos trabajando con el Hospital General de Ciudad Real. Muestra los participantes "Hospital Público Local", "Paciente" y "Registro Central de Salud", donde el paciente solicita programar la CMA (por ej. página Web), la secretaría del Hospital reserva día y hora tentativos que le son enviados al paciente (por ej. e-mail o sms) a la vez que se solicita el registro médico del paciente al Registro Central de Salud. El paciente se presenta el día asignado para la CMA entregando la orden para la cirugía, se chequean las precondiciones para la CMA (análisis de sangre, medicamentos, ayuno, etc.) y si hay algún problema se informa al paciente y se cancela la cirugía, de lo contrario se realizan los pasos definidos para preparar al paciente para la CMA.

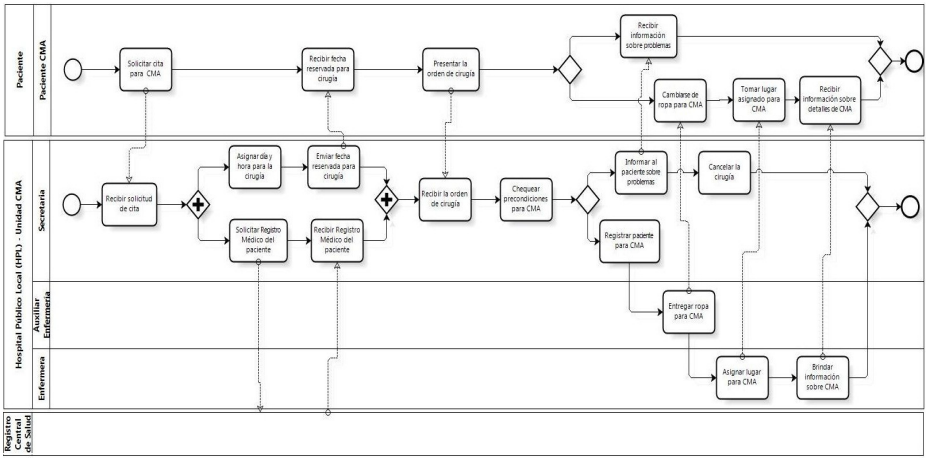


Fig. 1. Proceso "Admisión y Registro de paciente para CMA" en BPMN

2.2 Elementos clave en SoaML

El Lenguaje de Modelado para Arquitecturas Orientadas a Servicios (Service Oriented Architecture Modeling Language, SoaML) [9] de OMG provee un perfil UML y un metamodelo que extiende el metamodelo UML para diseñar servicios en SOA, la versión actual del estándar es la beta 2. En SoaML se define un servicio como una oferta de valor según una o más capacidades (abstracción de la habilidad de actuar y producir una salida y resultado) que tiene interface/s y un contrato asociados. Las interfaces pueden ser de tipo Interface de Servicio (*ServiceInterface*) o Interface simple (*Interface*) UML. Un Contrato de Servicios (*ServiceContract*) define los términos, condiciones, interfaces y coreografía en que los participantes acuerdan para utilizarlo, esta última puede expresar con cualquier diagrama de comportamiento (*Behavior*) UML, generalmente uno de secuencia. Una Arquitectura de Servicios (*ServiceArchitecture*) es una colaboración UML que presenta los participantes, contratos de servicios y roles en los mismos, brindando una visión global de los servicios provistos y requeridos. Los Participantes (*Participants*) pueden ser componentes de software, organizaciones, o sistemas que proveen y usan éstos servicios, ofreciendo capacidades en puntos de servicio (*Service*) y requiriendo servicios en puntos de solicitud (*Request*), siendo ambos (puntos de servicios y solicitud) especializaciones de *Port* UML (en la versión anterior beta 1: *ServicePoint* y *RequestPoint*). Un canal de servicios (*ServiceChannel*) modela la comunicación entre proveedores y consumidores de servicios, el tipo de mensaje (*MessagesType*) especifica la información intercambiada. La Fig. 2 muestra uno de los diagramas de servicios de SoaML, el diagrama de Arquitectura de Servicios (*ServicesArchitecture, SA*) correspondiente al PN de la Fig. 1.

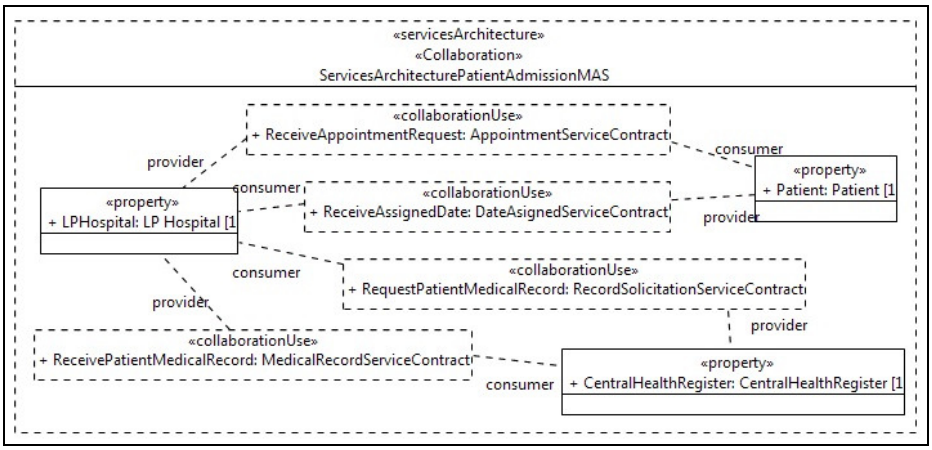


Fig. 2. Arquitectura de Servicios SoaML para el proceso de la Fig. 1

El diagrama de Arquitectura del modelo de servicios en la Fig. 2 muestra los participantes "Hospital PL", "Paciente" y "RegistroCentralSalud" que fueran definidos como pools en el PN en la Fig. 1. La Fig. 2 muestra también cuatro contratos de servicios (*ServiceContract*) correspondientes a los servicios provistos y requeridos por los participantes involucrados. Cada participante tiene un rol en la interacción con el servicio que puede ser proveedor y consumidor que se muestran en el diagrama. Esta conexión define también los puertos (*Service o Request*) donde los participantes proveen o consumen cada servicio. El diagrama de la Fig. 2, así como el resto de los diagramas de SoaML que se presentan en el artículo, están realizados con el plug-in SoaML para Eclipse que estamos desarrollando integrado en MINERVA.

2.3 Derivación y generación de servicios desde PN

La metodología BPSOM define como derivar servicios desde modelos conceptuales de PN, identificando los participantes involucrados y, para éstos, los servicios provistos y consumidos, contratos asociados con definición de interfaces, parámetros de entrada y salida y mensajes intercambiados. Con la aparición del estándar SoaML hemos incluido en la metodología su utilización en base a las definiciones previas realizadas, incluyendo en las actividades todos los elementos y diagramas de SoaML, y adecuándolas cuando se ha visto necesario. Adicionalmente al uso de BPMN y SoaML, definimos transformaciones QVT entre los metamodelos de dichos estándares, que nos permiten automatizar (parcialmente) la generación de servicios desde PN, obteniendo modelos de servicios desde modelos de PN. Nuestro enfoque para la generación de servicios en SoaML desde modelos de PN en BPMN sigue los principios de MDA y se basa completamente en el uso de estándares OMG. En la Fig. 3 se muestra la visión MDA propuesta por el marco MINERVA.

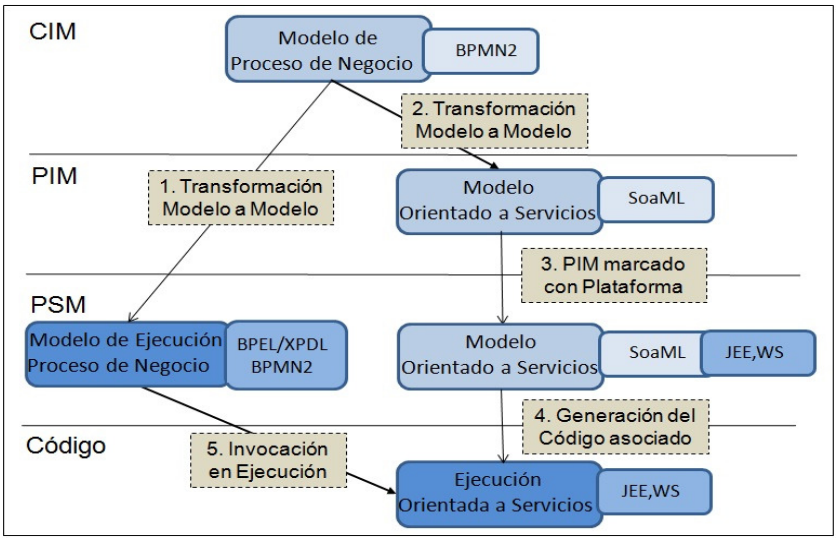


Fig. 3. Aplicación de MDA a MINERVA para generación de servicios desde PN

El modelo de PN en BPMN constituye el Modelo Independiente de la Computación (Computation Independent Model, CIM) en la transformación y el modelo de servicios en SoaML constituye el Modelo Independiente de la Plataforma (Platform Independent Model, PIM). Desde el modelo de servicios en SoaML se obtiene el Modelo Específico de la Plataforma (Platform Specific Model, PSM) en la plataforma seleccionada y el código asociado con motores MDA (ej. ModelPro[1]), lo que permite completar la trazabilidad desde el PN a su implementación. Los servicios serán luego invocados desde los motores de procesos ejecutando el PN en BPEL/XPDL/BPMN2 siendo el PSM y (código) ejecutable obtenido desde el modelo BPMN2 que con la transformación identidad es también el PIM (no incluido en la Fig.3). Las transformaciones QVT entre los metamodelos de los estándares (paso 2 Fig. 3) se basan en una ontología definida previamente que relaciona los elementos de modelado de PN y servicios [10], obteniendo luego las correspondencias de la Fig.4.

BPMN 2.0	SoaML beta2	BPMN 2.0	SoaML beta2
Definitions (complete model)	Model <Model>	Interface <Interface>	Interface <Interface>
Process	Participant <Class>	Operation <Operation>	Operation <Operation>
Message	MessageType <Class>	MessageRef In Out	Parameter In Out <Parameter>
ServiceTask (provider)	Service (of Participant) <Port>	Collaboration	Services Architecture <Collaboration>
Task (consumer)	Request (of Participant) <Port>	Collaboration Participant	Participant Part <Property>
MessageFlow sourceRef targetRef	ServiceContract <Collaboration> Provider <Property> provider Consumer <Property> consumer	MessageFlow sourceRef targetRef	CollaborationUse <Collaboration Use> Dependency consumer provider <Dependency>

Fig. 4. Correspondencias clave definidas entre elementos de BPMN2 y SoaML beta 2

[1] http://modeldriven.org/

3 Definición de BPSOM

BPSOM está definida para ser integrada en el proceso de desarrollo de software existente en una organización, ya que nuestro objetivo es reutilizar el conocimiento existente y solo agregar elementos específicos para guiar el desarrollo orientado a servicios (OS) desde PN. En [13] se presenta el proceso de definición de BPSOM, sus disciplinas, actividades, roles y artefactos, así como un ejemplo de su utilización. Desde esa definición hemos implementado la metodología como un *Method plug-in* en Eclipse Process Framework (EPF) Composer [15] y publicado como sitio Web [20], para proveer interoperabilidad con procesos definidos de la misma forma. A continuación extendemos la definición inicial de BPSOM introduciendo la de este artículo, dos elementos que consideramos clave para el desarrollo con servicios: SoaML para modelado de servicios, y la generación automática de modelos de servicios con transformaciones QVT desde BPMN2 a SoaML.

Las disciplinas que BPSOM define como prioritarias para el desarrollo orientado a servicios desde PN son: Modelado del Negocio, Diseño e Implementación. Cada disciplina incluye actividades clave para realizar la definición, especificación, diseño e implementación de servicios, así como roles involucrados, artefactos a producir (y requeridos) por las actividades y plantillas para su realización. Otras disciplinas de Ingeniería necesarias, como Requisitos, Verificación y Despliegue, son las propias del proceso de desarrollo existente, para poder reutilizar el proceso con el cual las personas están acostumbradas a trabajar. Lo mismo se define para las disciplinas de soporte como Aseguramiento de la Calidad, Gestión de la Configuración y Gestión del Proyecto. En la Fig. 5 se muestra la definición general de BPSOM y su uso en el contexto del proceso de desarrollo de software existente en una organización.

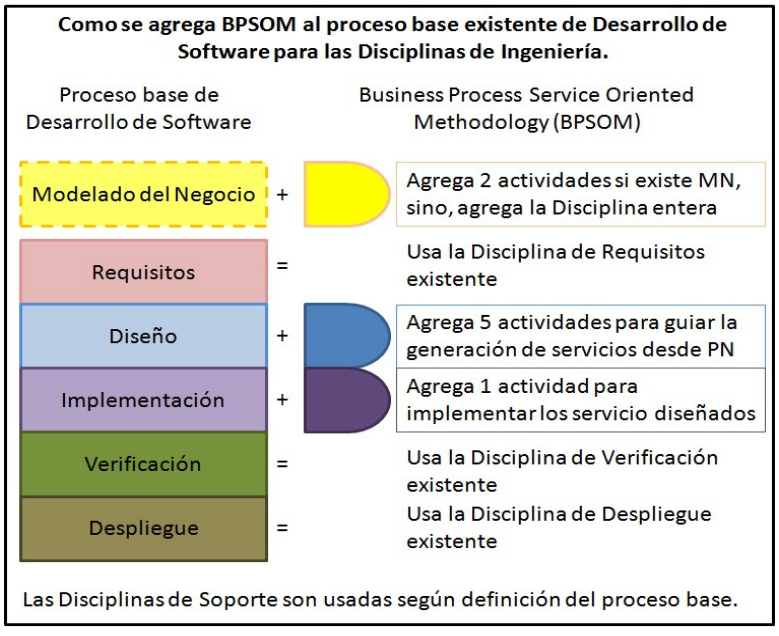


Fig. 5. Definición y uso de BPSOM en el contexto del proceso base de software existente

3.1 Disciplina Modelado del Negocio

La disciplina de Modelado del Negocio tiene como objetivo principal comprender el entorno organizacional e identificar los PN que definen su negocio, constituyendo uno de los pilares de BPSOM. En esta disciplina participan los Analistas del Negocio por el área del Negocio, y los Analistas de Sistemas y Arquitecto por el área de software.

BM1 – Evaluar la Organización objetivo. Tiene como objetivo involucrar al equipo de proyecto con la organización para la cual se realiza el desarrollo, si el área de software es de la misma organización entonces se podrán reutilizar documentos del negocio existentes donde se describa el área del negocio, los objetivos, su operación, empleados, tecnologías, entre otros. Adicionalmente el Modelo de Motivación del Negocio (Business Motivation Model, BMM) [21] de OMG, se utiliza para expresar los objetivos y estrategia del negocio y otra información del negocio relevante, que luego pueda ser relacionada directamente con la definición de servicios en SoaML.

BM2 – Identificar los Procesos de Negocio. Esta actividad es una de las actividades clave para el desarrollo de servicios desde PN, ya que constituye la entrada principal para comprender y describir los PN de la organización, principalmente los relacionados con la aplicación en desarrollo. Se utiliza BPMN como notación para especificar los modelos de PN, Es posible especificar los PN en otras notaciones pero las transformaciones QVT desarrolladas no podrían ser usadas. Para el modelado de PN recomendamos el uso de los patrones de procesos (*Workflow patterns*) [22] que proveen análisis y soluciones a problemas comunes de modelado. Un aspecto importante a especificar son los límites del negocio, indicando quién y qué interactúa con la organización, así como los puntos de interacción con el resto de las partes involucradas. En general toda la información puede ser especificada en los modelos de PN utilizando alguna herramienta de modelado BPMN (como BizAgi). Imágenes de los modelos se pueden incluir también en el documento de PN para su comunicación, donde se describen si es necesario, los PN en lenguaje natural para proveer una guía del modelado e incluir detalles de partes específicas del modelo que se quieran aclarar. Un ejemplo de modelo de PN en BPMN es el mostrado en la Fig.1

3.2 Disciplina de Diseño

La disciplina de Diseño es el otro pilar de la metodología BPSOM. Su objetivo principal es la definición y especificación de servicios desde PN, así como el reuso de los servicios existentes en la organización. En esta disciplina participan como responsable principal el Arquitecto, y los Analistas de Sistemas y Desarrolladores como soporte al diseño y especificación de los servicios.

D1 – Identificar y categorizar servicios. En esta actividad se identifican los servicios necesarios para realizar los PN en desarrollo, teniendo como principales entradas los PN especificados previamente y el Documento de la Arquitectura de Software (SAD) definido según el proceso de desarrollo de software existente. Los servicios que la organización debe proveer y los servicios que la organización

requiere consumir de otras partes involucradas, se identifican en base a los mensajes que se intercambian, cada parte definida por una calle (*Pool*) en el PN. Los servicios internos a la organización se identifican dentro de los carriles (*Lane*) que presenta cada calle, que serán luego modelados dentro de cada organización. En esta actividad se modela con SoaML la Arquitectura de Servicios (ServicesArchitecture, SA) especificando los participantes, contratos de servicios y los roles que cada participante cumple como proveedor o consumidor del servicio. La definición de servicios se realiza en alto nivel mostrando la visión general de servicios necesarios y los participantes que conectan, para todas las organizaciones involucradas. La Fig. 2 muestra el ejemplo del diagrama de SA en SoaML para el PN presentado en la Fig. 1.

Las calles del modelo de PN (definición de *Pool* en BPMN) se corresponden con Participantes en SoaML, luego para cada carril (*Lane*) dentro de la calle se generan participantes internos a éstos, en cada organización al refinar los modelos. Los servicios se identifican con las actividades identificadas para soportar servicios, a las que se les indica su tipo como *Service*, y con los mensajes intercambiados entre las calles. Teniendo en cuenta la dirección de los mensajes definimos que la actividad, en la cual el mensaje es entrante será soportada por el servicio asociado, y la actividad para la cual el mensaje es saliente será la que lo consuma. La calle que contiene cada actividad asociada con los servicios identificados define el participante que lo provee o lo requiere, dependiendo de si la actividad es soportada por un servicio provisto o requiere invocar uno. Esta información se incluye en el contrato del servicio (*ServiceContract*) para indicar los roles de proveedor y consumidor definidos. Para generar el diagrama de Arquitectura desde el modelo de PN, el arquitecto tiene que especificar en el modelo de PN que actividades relacionadas mediante mensajes de entrada y salida son de tipo *Service*, marcándolas con el estereotipo de BPMN, ya que otras actividades podrán ser de tipo manual o no soportadas por servicios, por lo que no serán tenidas en cuenta para la generación de los servicios requeridos. Junto con el diagrama de Arquitectura es se genera también el diagrama de participantes del PN incluyendo sus puertos y los servicios provistos y requeridos (*Service, Request*) según la identificación realizada. En la Fig. 6 se muestran los participantes del PN de ejemplo, así como sus puertos con servicios provistos y requeridos.

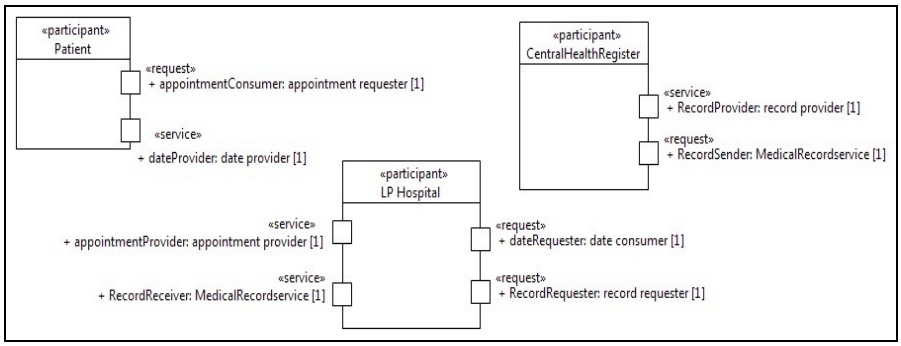


Fig. 6. Participantes con sus puertos y servicios provistos y requeridos

D2 – Especificar servicios. La especificación de servicios refiere a la definición de toda la información necesaria para su posterior implementación. Esto incluye definir toda la información del contrato de servicio (*ServiceContract*): interfaces, operaciones, parámetros de entrada y salida, pre y post condiciones si existieran, y la coreografía asociada al intercambio de mensajes por las partes involucradas. Esta información puede modelarse una vez generados los diagramas anteriores en base a las definiciones del servicio, o puede ser generada parcialmente desde el modelo de PN. Para esto último el arquitecto debe incorporarla al modelo de PN para aquellas actividades que fueron marcadas de tipo *Service*. La información relacionada con los mensajes de entrada y salida tiene que ser especificada (por ej. en los mensajes intercambiados), indicando los parámetros y tipos a ser intercambiados entre las partes. Es posible entonces definir los elementos *MessageType* a ser usados como parámetros en las operaciones, como se muestra en la Fig. 7.

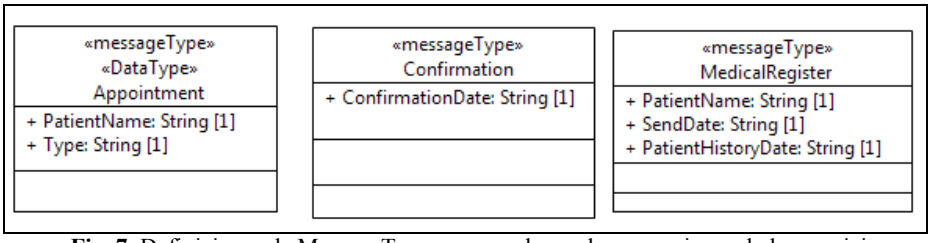


Fig. 7. Definiciones de MessageType a ser usados en las operaciones de los servicios

El contrato del servicio (*ServiceContract*) puede tener asociada también una coreografía para mostrar la interacción entre los participantes, interfaces, operaciones y parámetros utilizados. Teniendo en cuenta esta información para cada servicio, se generan las partes más importantes del contrato de servicio asociado, que son completadas posteriormente por el arquitecto o por los desarrolladores, que también tendrán que completar los detalles de implementación del servicio generado. Los principales elementos correspondientes al contrato de servicio para el servicio "RecibirSolicitudCita" se muestran en la Fig.8: la definición de los roles proveedor y consumidor y la asociación entre estos, así como su coreografía asociada. Se definen también las interfaces, operaciones y parámetros para el servicio, que no se muestran en detalle por razones de espacio.

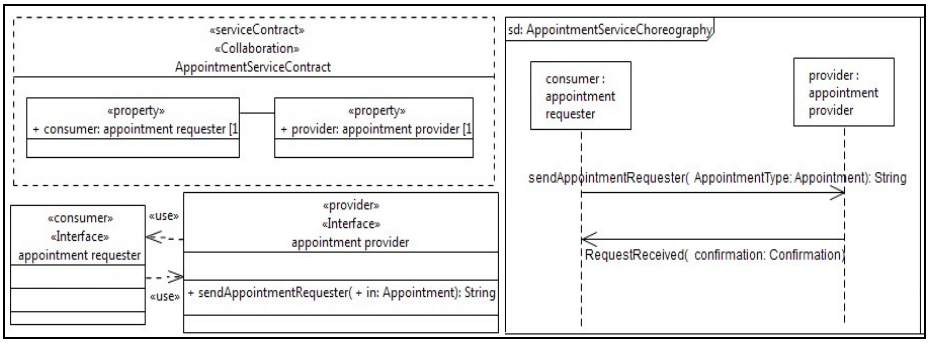


Fig. 8. Especificación del servicio "ReceiveAppointmentRequest"

D3 – Investigar servicios existentes. Esta actividad tiene como principal objetivo reusar lo más posible los servicios existentes en la organización, para lo cual se define un Catálogo de Servicios centralizado donde cada nuevo servicio que se desarrolla es registrado, y en el cual para cada proyecto se buscan servicios que provean la funcionalidad requerida o similar. En ambos casos, en un diagrama de componentes de SoaML se define un componente de tipo *adapter* o *wrapper* para relacionar el servicio generado en el modelo de servicios con su implementación existente.

D4 – Asignar servicios a componentes. Los componentes que implementarán los servicios generados serán definidos y presentados en el diagrama de componentes. SoaML provee el componente participante para definir la implementación de participantes y servicios, definiendo nuevos componentes a generar, y si los componentes existen, la definición de *adapters* o *wrappers* para usarlos, proveyendo las modificaciones necesarias para su invocación.

D5 – Definir la interacción de servicios. La interacción de servicios puede definirse como orquestación o coreografía de servicios de la misma forma que fue definida para los PN. Un diagrama de secuencia mostrando todos los servicios, o varios diagramas de secuencia mostrando distintos subconjuntos de servicios correspondientes a distintos subprocesos en el PN, proveen esta información. Esta actividad no tiene diagrama correspondiente en SoaML y se realiza mediante el diagrama de secuencia UML, en forma manual por el arquitecto.

3.3 Transformaciones QVT

El algoritmo general se compone de varias relaciones QVT que se van invocando a medida que los elementos necesarios van siendo generados. A modo de ejemplo en la Fig. 9 se muestra la generación de la especificación de servicios en notación gráfica de QVT, donde la conexión entre los estereotipos SoaML aplicados a los elementos base UML no se indica ya que está implícita en las referencias de los atributos.

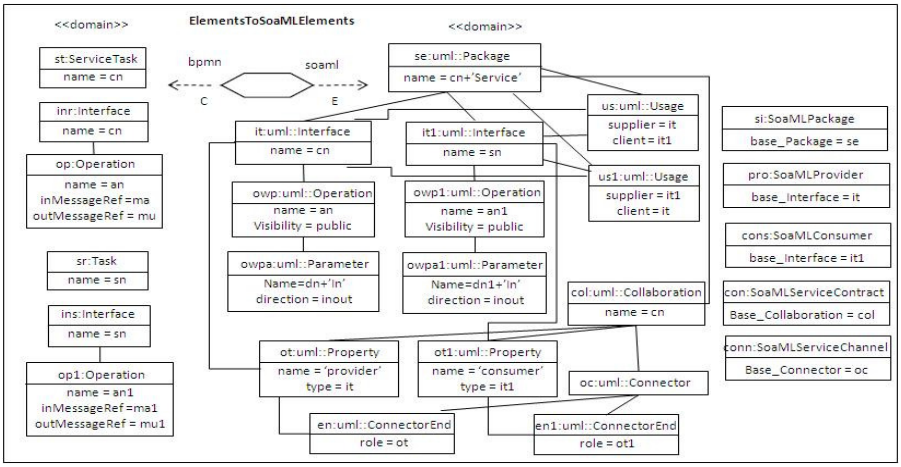


Fig. 9. Una de las relaciones QVT para la generación de servicios en notación gráfica QVT

4 Trabajos relacionados

En los últimos años se han definido varias metodologías para desarrollo de servicios, contemporáneamente con BPSOM, tanto desde el área académica como desde el área industrial, las que se mencionan en [13], y se discuten también en [18]. Entre ellas, se pueden mencionar la propuesta de Papazoglou et. al [23] que define una metodología para diseño y desarrollo con servicios desde PN, para desarrollo específico con WS, con seis fases realizadas iterativamente: planificación, análisis y diseño, construcción y testing, provisionamiento, despliegue, ejecución y monitoreo. El plug-in SOMA [24] de RUP para definir servicios de negocio y software en una metodología consolidada, como en nuestra propuesta, agrega elementos específicos pero al proceso RUP, y la metodología del proyecto SHAPE [25] que también integra SoaML pero con distintas guías de desarrollo que nuestra propuesta y sin generación automática. Kohlborn et al. [26] propone un enfoque consolidado que combina definiciones de los trabajos de desarrollo con servicios revisados, agregando también nuevos ítems. Define dos partes principales para el proceso: la derivación de servicios del negocio y la derivación de servicios de software para soportarlos.

En [27] Herold et. al propone un enfoque dirigido por modelos con cuatro fases: desarrollo del negocio, análisis de requisitos, diseño arquitectónico y modelado de implementación, con guías y transformaciones para moverse de uno modelo al otro. De Castro et al. [28] define un método para composición de servicios con un proceso que establece varios pasos para la generación de modelos, definiendo metamodelos, modelos y artefactos a obtener. En [29] de Castro et. al integran un modelo de valor del negocio para derivar artefactos desde el mismo usando ATL [30]. Este lenguaje para transformaciones también es utilizado en Touzi et. al [31] donde se definen modelos, metamodelos y transformaciones para ir desde un PN colaborativo (CIM) a un modelo SOA (PIM) generando el código en BPEL en base a PIM4SOA. Nuestra propuesta difiere de las anteriores en varios sentidos. En primer lugar la definición de la metodología BPSOM para guiar el desarrollo de servicios desde PN se agrega al proceso base existente en la organización, facilitando su adopción. En segundo lugar las transformaciones se definen y ejecutan con QVT en el mismo entorno de desarrollo, obteniendo los modelos de servicios en SoaML directamente desde modelos de PN en BPMN2, sin generar ni utilizar ningún artefacto intermedio. Desde los modelos SoaML a su vez, es posible generar el código correspondiente integrando motores MDA en el mismo entorno. En tercer lugar, la guía conceptual y automática está integrada y detallada claramente en la metodología BPSOM, haciéndola fácilmente accesible y utilizable en el entorno de desarrollo de software.

5 Conclusiones y trabajo futuro

La metodología BPSOM está definida para guiar desarrollos orientados a servicios desde procesos de negocio, siendo integrada en el marco MINERVA para mejora continua de PN, en el que estamos trabajando desde 2009. BPSOM se ha pensado para ser agregada al proceso de desarrollo existente en la organización, agregando únicamente los elementos clave para desarrollo con servicios. En trabajos previos mencionados se ha presentado su implementación como *Method plug-in* de EPF

Composer y publicada como sitio web para su interoperabilidad con otros procesos definidos en la misma forma. En este artículo hemos presentado la integración del estándar SoaML para modelado de servicios indicando en cada actividad de BPSOM que diagramas de SoaML deben ser realizados, brindando guías para su realización, así como los roles involucrados y sus responsabilidades. La definición de servicios se va completando en forma iterativa e incremental desde la definición de la Arquitectura de Servicios hasta la especificación completa de los mismos, incluyendo interfaces, operaciones, parámetros asociados y mensajes intercambiados. Asimismo, hemos incluido en BPSOM la generación automática de modelos de servicios en SoaML desde modelos de PN en BPMN2 mediante transformaciones QVT desde los metamodelos de BPMN a SoaML, que se ejecutan en el entorno Eclipse definido por MINERVA. Recientemente hemos completado las transformaciones QVT que permiten obtener modelos SoaML completos y que además pueden ser visualizados en el plug-in SoaML[2] de Eclipse que hemos desarrollado, en el cual están modelados todos los diagramas SoaML presentados en este artículo. A partir de los modelos SoaML es posible generar el código asociado mediante motores MDA existentes, cuya integración en el entorno definido es uno de los trabajos que estamos realizando actualmente. Esto permite cerrar el ciclo de modelado–generación–diseño-implementación propuesto. Asimismo estamos trabajando en un caso real de PNs del Hospital General de Ciudad Real para la validación de la propuesta, desde el modelado de PN en BPMN, la generación de modelos de servicios en SoaML, hasta la ejecución de los PN en motores de procesos invocando los servicios generados.

Agradecimientos. Este trabajo ha sido financiado parcialmente por la Agencia Nacional de Investigación e Innovación (ANII,Uruguay), proyecto ALTAMIRA (Junta de Comunidades de Castilla-La Mancha, España, Fondo Social Europeo, PII2I09-0106-2463), proyecto PEGASO/MAGO (Ministerio de Ciencia e Innovacion MICINN, España, Fondo Europeo de Desarrollo Regional FEDER, TIN2009-13718-C02-01), proyecto INGENIOSO (Junta de Comunidades de Castilla-La Mancha, España, PEII11-0025-9533) y proyecto MOTERO (Junta de Comunidades de Castilla-La Mancha, España, PEII11-0366-9449)

Referencias

1. Weske, M. 2007. BPM Concepts, Languages, Architectures, Springer.
2. Smith, H., Fingar, P. 2003. Business Process Management: The third wave, Meghan-Kieffer.
3. van der Aalst, W.M.P., ter Hofstede, A., Weske, M., Business Process Management: A Survey, In: International Conference on Business Process Management, (2003)
4. Papazoglou, M., Traverso, P., Dustdar, S. and Leymann, F. Service-Oriented Computing: State of the Art and Research Challenge, IEEE Computer Society, (2007)
5. Krafzig, D., Banke, K., Slama, D.,Enterprise SOA,SOA:Best Practices, Prentice Hall, (2005)
6. Erl, T., SOA: Concepts, Technology, and Design, Prentice Hall, (2005)
7. Mellor, S., Clark, A., Futagami, T., MDD, Guest editors intro, IEEE CS, (2003)
8. Object Management Group (OMG), Model Driven Architecture (MDA), (2003)
9. Object Management Group (OMG), SOA Modeling Language (SoaML), (2009)

[2] http://alarcos.esi.uclm.es/MINERVA/TOOLS/

10. Delgado A., Ruiz F., García-Rodríguez de Guzmán I., Piattini M:, MINERVA: Model drIveN and sErvice oRiented framework for the continuous business processes improVement & relAted tools, 5th Int. Work. on Engineering Service-Oriented Applications (WESOA'09), with ICSOC 2009, Estocolmo, Noviembre (2009).
11. Delgado A., Ruiz F., García-Rodríguez de Guzmán I., Piattini M: A Model-driven and Service-oriented framework for the BP improvement, Journal of Systems Integration (JSI), Vol.1, No.3, ISSN: 1804-2724, Julio (2010).
12. Delgado A., Ruiz F., García-Rodríguez de Guzmán I., Piattini M., Towards an ontology for service oriented modeling supporting business processes, 4th. International Conference on Research Challenges in Information Science (RCIS'10), Nice, Mayo (2010).
13. Delgado A., Ruiz F., García-Rodríguez de Guzmán I.,Piattini M., Towards a Service-Oriented and Model-Driven framework with business processes as first-class citizens, In: 2nd IC on Business Process and Services Computing (BPSC 09),Leipzig, Marzo (2009)
14. Delgado A., Ruiz F., García-Rodríguez de Guzmán I.: Mejora continua de procesos de negocio basada en PmCOMPETISOFT integrando BPMM. III Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS'10) en XV JISBD'10, Valencia, Setiembre (2010)
15. Delgado, A., García - Rodríguez de Guzmán, I., Ruiz, F., Piattini, M.: Tool support for Service Oriented development from Business Processes, 2nd International Workshop on Model-Driven Service Engineering (MOSE'10), Málaga, Junio (2010)
16. Object Management Group (OMG), Query/Views/Transformations(QVT), (2008)
17. Object Management Group (OMG), Business Process Modeling Notation (BPMN2), (2011)
18. Delgado A., Ruiz F., García-Rodríguez de Guzmán I., Piattini M, Application of service-oriented computing and model-driven development paradigms to business processes: a systematic review, 5th Int. Conf. on Software and Data Techs. (ICSOFT'10), Atenas, (2010)
19. Delgado, A., García - Rodríguez de Guzmán, I., Ruiz, F., Piattini, M: From BPMN business process models to SoaML service models: a transformation-driven approach. 2nd Int.Conf. on Software Tech. and Engineering (ICSTE 2010), San Juan de Puerto Rico, Octubre (2010)
20. Delgado, A., BPSOM Methodology http://alarcos.esi. uclm.es/MINERVA/BPSOM/ (2010)
21. Object Management Group (OMG), Business Motivation Model (BMM), (2010)
22. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14, 5–51, (2003)
23. Papazoglou, M., van den Heuvel, W., Service-oriented design and development methodology, Int. J. Web Engineering and Technology, Vol. 2, No. 4, pp.412-462, (2006)
24. IBM-SOMA, < http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/>
25. Stollberg, M., et al, A Customizable Methodology for the MDE of Service-based System Landscapes. 4th Workshop on Modeling, Design, and Analysis for the Service Cloud (MDA4ServiceCloud'10), with ECMFA'10, Paris, June, (2010)
26. Kohlborn T., Korthaus A., Chan T., Rosemann M., Identification and Analysis of Business and SE Services- A Consolidated Approach, IEEE Transactions on Services Comp., (2009)
27. Herold S., Rausch A., Bosl A., Ebell J., Linsmeier C., Peters D., A Seamless Modeling Approach for Service-Oriented Information Systems, In ITNG'08, 5th International Conference on Information Technology:New Generations, (2008)
28. de Castro, V., Marcos, E., López Sanz, M., A model driven method for service composition modelling: a case study, Int. J. Web Engineering and Technology, Vol. 2, No. 4, (2006)
29. de Castro V., Vara Mesa J. M., Herrmann E., Marcos E., A Model Driven Approach for the Alignment of Business and Information Systems Models, (2008)
30. Jouault, F., Kurtev, I.,Transforming Models with ATL (ATLAS Transformation Language), Satellite Events at the MoDELS Conference, (2005)
31. Touzi J., Benaben F., Pingaud H., Lorré J.P., A model-driven approach for collaborative service-oriented architecture design, Int. Journal of Prod. Economics, Vol.121 Is.1, (2009)

Sesión 5: Procesos de negocio

Procesos de Negocio Auto-Adaptables al contexto*

Clara Ayora[1], Germán H. Alférez[2], Victoria Torres[1], and Vicente Pelechano[1]

[1] Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain
{cayora, vtorres, pele}@pros.upv.es
[2] Facultad de Ingeniería y Tecnología
Universidad de Montemorelos
Apartado 16-5 Montemorelos N.L. México
harveyalferez@um.edu.mx

Abstract. Cuando los procesos de negocio se ejecutan en entornos altamente dinámicos, los continuos cambios de contexto que se dan en dichos entorno obligan a adaptar constantemente estos procesos. Llevar a cabo esta adaptación de forma manual no es factible debido a la complejidad y al alto coste que esto conlleva. Por lo tanto se requiere de técnicas más avanzadas basadas en auto-adaptación. Este tipo de técnicas las encontramos en la Computación Autónoma, la cual ofrece la posibilidad de reconfigurar sistemas en base a cambios en el contexto. Es por esto que en este trabajo se presenta una propuesta para manejar y adaptar automáticamente procesos de negocio variables de acuerdo a cambios en el contexto. Específicamente, la propuesta permite 1) modelar el proceso previniendo variaciones en el contexto de ejecución y 2) adaptar dicho proceso en función de estas variaciones. Además, se propone la infraestructura software que permite esta adaptación en tiempo de ejecución.

Keywords: Procesos de Negocio, Reconfiguración de Procesos de Negocio, Modelos en Tiempo de Ejecución

1 Introducción

Un proceso de negocio está formado por una composición de tareas relacionadas cuya ejecución logra un objetivo específico [1]. Llevar a cabo una gestión adecuada de estos procesos permite a las organizaciones no sólo reducir esfuerzos, sino además, obtener ventajas competitivas frente a otras organizaciones. Sin embargo, esta gestión no es una tarea fácil de realizar y menos aún cuando los procesos se llevan a cabo en entornos altamente dinámicos, los cuales conllevan

* Este trabajo ha sido desarrollado con el apoyo del MICINN dentro del proyecto EVERYWARE TIN2010-18011.

un alto nivel de variabilidad. Cómo adaptarse y soportar estos cambios del entorno resulta clave para el éxito de las organizaciones. Aunque se han realizado numerosos esfuerzos ([2], [3], [4]) para proveer de mecanismos que permitan llevar a cabo esta gestión de la variabilidad, éstos no llevan a cabo esta gestión utilizando el propio modelo del proceso. Esto fuerza a contemplar y desplegar los procesos de forma global y completa, lo cual es realmente complicado debido al tamaño y a la complejidad de los procesos que se manejan en la actualidad. Gracias a la utilización de la Computación Autónoma [5] (concepto introducido por IBM), se puede reducir esta complejidad mediante el uso de mecanismos de auto-adaptación (también llamados de auto-reconfiguración). Con este objetivo, en este trabajo se presenta una propuesta para definir, ejecutar y adaptar automáticamente los procesos de negocio que varían en función del contexto de ejecución. Esta propuesta cubre desde el modelado del proceso hasta cómo éste debe comportarse cuando se producen cambios en su entorno de ejecución.

Concretamente, la propuesta cubre dos de las fases del ciclo de gestión de los procesos de negocio las cuales se refieren al diseño y a la ejecución. Por un lado, durante la fase de diseño se definen los modelos de proceso que representan el sistema. Dichos modelos representan también las diferentes variantes del proceso de acuerdo a los diferentes contextos en los que el proceso puede verse envuelto desde su puesta en ejecución hasta su finalización. Para llevar a cabo dicha definición, en este trabajo se propone la utilización del *Common Variability Language* [6] (*CVL*), estándar en desarrollo para representar la variabilidad en un determinado sistema. CVL permite dotar a cualquier lenguaje específico de dominio la suficiente expresividad para definir la variabilidad que pueda existir en dicho dominio. En nuestro caso, este dominio se refiere a los procesos de negocio y el lenguaje utilizado podría ser cualquier lenguaje de modelado de procesos como por ejemplo EPC, YAWL, BPMN, diagramas de actividad de UML, etc. En particular, en este trabajo se utiliza BPMN, específicamente la versión 2.0, la cual, además de ser el estándar propuesto por la OMG para la definición de procesos, está dotada de capacidad de ejecución, hecho que facilita el despliegue de dichos procesos.

Por otro lado, durante la fase de ejecución, para llevar a cabo reconfiguraciones de los procesos de acuerdo a cambios en el contexto de ejecución, se propone el uso de técnicas de reconfiguraciones automáticas basadas en modelos en tiempo de ejecución. Concretamente, sobre el motor de ejecución llamado *Model-based Reconfiguration Engine for Business Processes* (MoRE-BP) los modelos del proceso de negocio puede ejecutarse y reconfigurarse automáticamente de acuerdo a cambios producidos en el contexto. Estas reconfiguraciones se llevan a cabo a través de un *plan de reconfiguración* encargado de modificar la composición de tareas que forman la ejecución del propio proceso de negocio.

El resto de este trabajo está organizado de la siguiente manera: en la sección 2 se describe un caso de estudio que se utiliza en las siguientes secciones para presentar la propuesta. La sección 3 presenta una descripción detallada de la propuesta describiendo cada uno de los modelos incluidos en ella así como la metodología seguida para su construcción, ejecución y reconfiguración

durante la fase de ejecución. En la sección 4 se presenta una recopilación de las propuestas existentes para la gestión de la variabilidad en los procesos de negocio. Finalmente, en la sección 5 se presentan las conclusiones y las direcciones del trabajo futuro.

2 Caso de estudio

Para ejemplificar la propuesta presentada en este trabajo, se presenta una versión simplificada del proceso que se lleva a cabo en una agencia de viajes para expedir billetes de avión. Este caso se centra en la variabilidad que existe relacionada con la tramitación de visados al país de destino. La gran casuística que existe en este tipo de sistemas es muy alta debido al número de países contemplados, tanto de origen como de destino. Por ejemplo, los ciudadanos colombianos no requieren visado para entrar a Singapur, mientras que sí lo requieren para entrar a España. Además, esta casuística varía continuamente en función de los diferentes acuerdos establecidos entre los diferentes países.

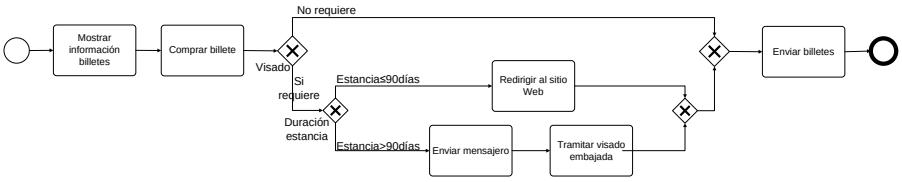


Fig. 1. Diagrama BPMN para la expedición de billetes y tramitado de visados

La figura 1 muestra una descripción gráfica de dicho proceso en BPMN. El proceso comienza cuando el usuario decide reservar un billete de avión a un país concreto. En primer lugar, el usuario introduce los datos en el sistema con el objetivo de localizar los billetes que más se ajustan a sus necesidades. Para ello, el sistema debe ofrecerle toda la información detallada de cada vuelo disponible a ese país. A continuación, el usuario realiza la compra del billete. Seguidamente, el sistema le ofrece la opción de ayudarle con los trámites para el visado correspondiente al país de destino. Es en este punto cuando el proceso puede cambiar dependiendo de las características del país de destino. Si el país de destino no requiere la expedición de visado, el sistema no realiza ninguna tramitación. Sin embargo, si por el contrario el país de destino requiere de visado, las tareas a realizar por el sistema dependerán de la duración de la estancia del usuario en el país de destino. Si el usuario va a permanecer menos de 90 días en el país, el visado puede ser tramitado de manera *on-line*, por lo que el sistema redirigirá al usuario al sitio web correspondiente para dicha tramitación. En cambio, si el usuario va a permanecer más de 90 días, es necesario realizar la tramitación del visado directamente en la embajada del país de destino. Para

facilitarle este proceso al usuario, el sistema ofrece la posibilidad de encargarse de esta gestión. Para ello, la agencia (1) envía un mensajero, que recogerá la documentación del usuario (pasaporte y dos fotografías), y (2) se encarga de realizar los trámites necesarios en la embajada correspondiente. Por último, una vez finalizado el proceso de obtención del visado, sea *on-line* o bien *in situ* en la embajada, el sistema envía los billetes de avión al usuario.

3 Propuesta para gestionar procesos de negocio auto-adaptables al contexto

En esta sección se presentan los detalles de la propuesta presentada para diseñar procesos de negocio capaces de adaptarse a los cambios que ocurren debido a su contexto de ejecución. Tal y como se muestra en la figura 2, la propuesta se desarrolla sobre dos fases del ciclo de vida de los procesos de negocio: (1) el análisis y diseño, donde se definen los modelos de proceso y (2) la ejecución, donde se lleva a cabo una reconfiguración de los modelos en tiempo de ejecución de acuerdo a cambios que ocurren en el contexto.

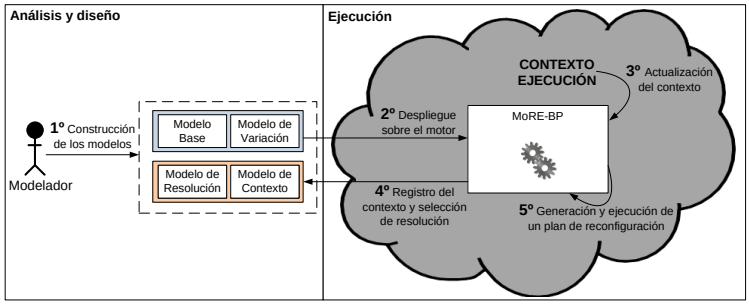


Fig. 2. Metodología de la propuesta presentada

En la primera fase se construyen los modelos que recogen la información relevante para la reconfiguración del proceso en tiempo de ejecución. Específicamente, se construyen: el Modelo Base, el Modelo de Variación, el Modelo de Resolución y el Modelo de Contexto (paso 1 de la figura 2).

El objetivo de la segunda fase es ejecutar y reconfigurar el proceso en función de los cambios producidos en el contexto. Específicamente, al comienzo de esta fase, el Modelo Base y el Modelo de Variación definidos en el paso 1 se despliegan sobre un motor de ejecución de procesos (MoRE-BP) para que comience su ejecución (paso 2). Durante esta ejecución, el motor recibe continuamente los cambios que se produzcan en el contexto (paso 3). Cada vez que se produce un cambio, MoRE-BP inserta la nueva información del contexto en el Modelo de Contexto y comprueba si alguna de las resoluciones definidas en el Modelo de

Resolución da respuesta a ese cambio (paso 4). Si se encuentra una resolución, se genera y ejecuta un plan de reconfiguración para dictaminar qué cambios se deben aplicar en el modelo del proceso de negocio con el fin de adaptar el proceso al nuevo contexto de ejecución (paso 5).

3.1 Definir los modelos para el proceso de negocio en tiempo de análisis y diseño

En esta fase, el objetivo es definir los modelos necesarios para dar soporte a la variabilidad en los procesos de negocio. Estos modelos se construyen contemplando todas las posibles variaciones que puede sufrir el proceso según el contexto de ejecución. Así, cuando se produzca algún cambio en el contexto, al haber sido previsto, el proceso se puede adaptar de forma automática a su nuevo entorno. Identificar qué partes del proceso deben ser adaptadas y bajo qué condiciones resulta clave para llevar a cabo dicha adaptación.

En el ámbito de Líneas de Producto Software (SPL) encontramos técnicas adecuadas para poner en práctica esta idea. Específicamente, la aproximación Base-Variación-Resolución (BVR) [7] propone la construcción de tres modelos: (1) el Modelo Base, (2) el Modelo de Variación y (3) el Modelo de Resolución para definir de forma explícita las partes del modelo que son fijas, las que pueden cambiar y las condiciones que harán que éstas cambien. Para implementar esta aproximación, se utiliza *CVL*, el estándar en desarrollo para modelar la variabilidad [8]. Esta aproximación considera la variabilidad de forma ortogonal al lenguaje específico de dominio sobre el que es aplicada, por lo que la variabilidad puede ser desacoplada y aislada del resto del modelo. Intentar integrar todos los aspectos concernientes a la variabilidad dentro de un mismo modelo mediante condiciones lógicas en el flujo de control, resulta en modelos muy grandes[3], los cuales son difíciles de entender y caros de mantener [9]. Además, al utilizar estas condiciones lógicas, no es posible distinguir qué partes del modelo son variables de cuáles pertenecen a la lógica común del propio proceso [10]. Así, al utilizar *CVL* no sólo se minimiza el impacto que la variabilidad pueda tener sobre el modelo sino que, además, los modelos resultantes son más entendibles, flexibles y escalables. Además de estos tres modelos, se propone la definición de un Modelo de Contexto para recoger, formalizar y analizar toda la información relativa al contexto de ejecución del proceso.

1. El **Modelo Base** es el modelo donde se especifican los elementos comunes a todas las variantes del proceso. Además, también se identifican los puntos concretos del propio modelo (llamados *placements* en términos de *CVL*) que están sujetos a cambios. Estos *placements* representan cajas negras cuya instanciación puede ser resuelta tanto en tiempo de diseño (cuando la selección de una alternativa depende del contexto inicial del proceso, por ejemplo cuando el proceso va a ser configurado para un dominio específico)

[3] Los escenarios reales contienen multitud de variaciones

o en tiempo de ejecución (cuando la selección de una alternativa depende del contexto actual de la instancia del proceso). Esta diferenciación se hace explícita en el propio *placement* mediante los estereotipos: «tiempo-diseño» y «tiempo-ejecucion». Este trabajo se centra únicamente en los procesos de negocio cuyas composiciones de tareas pueden variar en tiempo de ejecución debido a cambios en el contexto, por lo que solamente se utilizan los *placement* de tipo «tiempo-ejecucion».

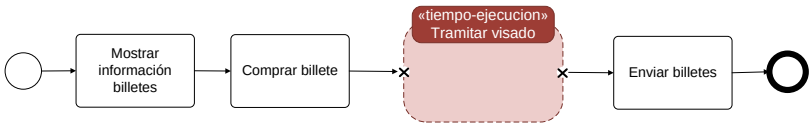


Fig. 3. Modelo Base del proceso para el caso de estudio

La figura 3 presenta el Modelo Base para el caso de estudio presentado en la sección 2. En ella se muestran (1) el conjunto de elementos comunes a todas las variantes del modelo (eventos de *inicio* y *fin*, y las tareas *Mostrar información billetes*, *Comprar billete* y *Enviar billete*) y (2) el *punto de variación* (*placement Tramitar visado*) cuya instanciación depende de las condiciones del contexto sobre el que se ejecuta el proceso (en particular, depende del país de destino y de la duración de la estancia en el país).

2. El **Modelo de Variación** es el modelo donde se especifican las diferentes alternativas (llamadas *replacements* en términos de *CVL*) que pueden encajar en los diferentes *placements* identificados en el Modelo Base. Para cada uno de ellos, se definen todos los *replacements* que pueden ser instanciados. A su vez, un *replacement* puede incluir nuevos *placements* de forma que se promueve la reusabilidad de los fragmentos del proceso.

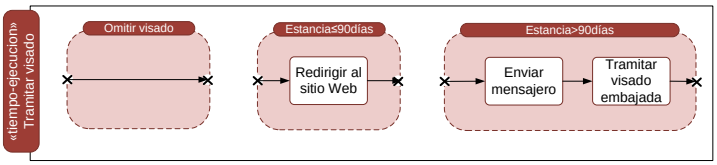


Fig. 4. Modelo de Variación del proceso para el caso de estudio

La figura 4 presenta el Modelo de Variación para el caso de estudio presentado en la sección 2. En ella se muestran las diferentes alternativas (representadas por *replacements*) que pueden instanciarse en el *placement*

Tramitar visado. Concretamente, la tramitación del visado puede (1) *omitirse* si el país de destino no requiere visado, (2) realizarse *on-line* si la estancia es menor o igual a 90 días, con lo que el sistema *Redirigirá al sitio Web* específico, o (3) el sistema *Enviará un mensajero* para poder *Tramitar el visado en el embajada* si la estancia es mayor de esos 90 días.

3. El **Modelo de Resolución** es el modelo donde se especifica el conjunto de *replacements* a seleccionar para un contexto en particular. Concretamente, esta especificación se construye identificando (1) los *fragmentos aplicables*, es decir, para cada *placement* definido en el Modelo Base, qué *replacements* deben ser seleccionados y (2) las *condiciones del contexto*, en términos de variables de contexto, que determinan esta selección. Así, si durante la ejecución del modelo se produce algún cambio en el contexto que coincide con alguna de las condiciones especificadas, el modelo deberá reconfigurar los *placements* indicados con los *replacements* especificados.

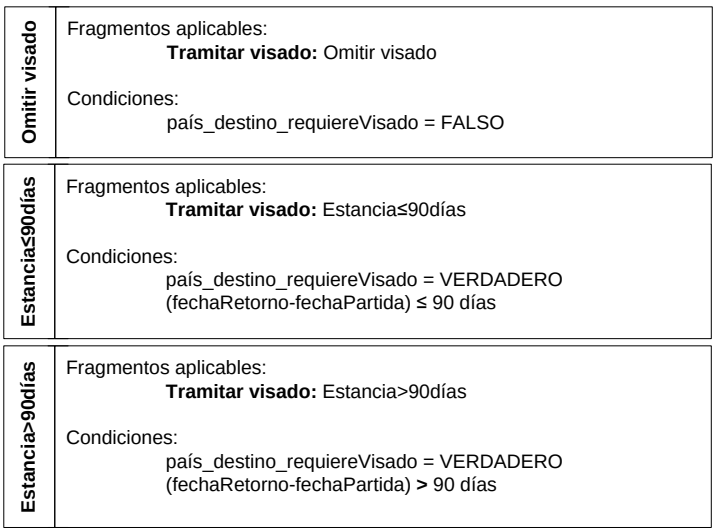


Fig. 5. Modelo de Resolución del proceso para el caso de estudio

La figura 5 presenta el Modelo de Resolución para el caso de estudio presentado en la sección 2. Concretamente, muestra el conjunto específico de resoluciones para el *placement Tramitar visado*. Estas resoluciones están estructuradas en dos bloques: (1) los *fragmentos aplicables*, donde se especifica la selección del *replacement* concreto (*Omitir visado*, *Estancia≤90días* y *Estancia>90días*) y (2) las *condiciones*, donde se especifica el conjunto de variables de contexto, y sus valores, que configuran está selección.

4. El **Modelo de Contexto** es el modelo donde se recogen todas las características (atributos) del contexto del proceso de negocio. Se entiende

por contexto de un proceso de negocio toda la información externa que puede afectar al proceso en tiempo de ejecución. Esta información puede ser introducida de forma manual o automática. Un ejemplo de introducción manual ocurre cuando el usuario envía una respuesta al sistema con el país de destino a viajar. Por otra parte, el navegador puede enviar automáticamente al sistema la información de la localización del usuario con el fin de presentarle información en su idioma. El mayor beneficio de utilizar un Modelo de Contexto es que éste permite realizar un análisis formal del propio contexto utilizando lógica de primer órden. La construcción de este modelo de contexto se basa en la *Web Semántica* [11] y en *OWL* (*Web Ontology Language*) [12]. Básicamente, *OWL* es un lenguaje de marcado de ontologías que permite compartir información contextual y razonar acerca del contexto. Específicamente, en la creación de la ontología se ha utilizado *Protégé-OWL* [13]. Como trabajo en curso, estamos estudiando la posibilidad de ver al contexto como un espacio de información delineado a lo largo de múltiples dimensiones. Estas dimensiones representan aspectos que en conjunto caracterizan al contexto del proceso de negocio [14].
La figura 6 presenta el Modelo de Contexto para el caso de estudio descrito en la sección 2 (esta figura ha sido construída con la herramienta *OntoGraf*[4]). La clase *Billete* que contiene la información que el usuario ha introducido para comprar un billete. La clase *País* que especifica si el país de destino requiere visado o no. Cada una de estas clases es interpretada como grupos de individuos. Concretamente, en la figura se muestran dos individuos de la clase *Billete* y tres individuos que representan a tres países diferentes. Además, también se especifican las relaciones existentes entre los individuos, por ejemplo la relación *Tiene destino* entre el individuo *Billete2* y el individuo *España*. Por otra parte, cada individuo de la clase *Billete* tiene una propiedad interna *de tipo dato* que determina la duración de la estancia del usuario en el país de destino (*fechaRetorno-fechaPartida*)[5]. En el caso del *Billete1* esta propiedad es de 95 días y en el caso del *Billete2* es de 3 días. Por otra parte, cada país tiene una *propiedad de tipo de dato* con el valor *NecesitaVisado*, que es de tipo booleano. Por ejemplo, en el caso del individuo *Australia*, esta propiedad tiene un valor booleano falso. Todas estas *propiedades de tipo dato* son las que, al variar su valor en tiempo de ejecución, hacen variar el proceso determinando qué *replacements* deben seleccionarse.

3.2 Ejecución y reconfiguración del modelo en tiempo de ejecución

En esta sección se presenta la estrategia a seguir para ejecutar el proceso de negocio y conseguir su auto-adaptación al contexto de ejecución de acuerdo a cambios que puedan producirse. Esta estrategia está basada en el uso de modelos en tiempo de ejecución.

[4] `http://protegewiki.stanford.edu/wiki/OntoGraf`
[5] la figura 6 no muestra las propiedades internas *de tipo dato* ya que *OntoGraf* no soporta su visualización

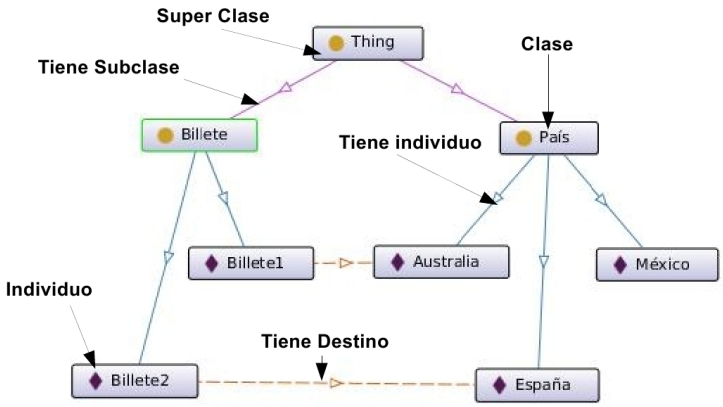


Fig. 6. Modelo de Contexto del proceso para el caso de estudio

Ejecución del modelo del proceso: Una vez se han construido los modelos necesarios para representar el proceso, puede comenzar su ejecución. Concretamente, el Modelo Base y el Modelo de Variación se despliegan sobre el motor MoRE-BP quien llevará a cabo su ejecución. MoRE-BP es una extensión de *Model-Based Reconfiguration Engine* (MoRE) [15], un motor de ejecución que permite la reconfiguración de modelos en tiempo de ejecución. MoRE está basado en el marco de trabajo OSGi [16]. Específicamente, OSGi permite instalar, iniciar, parar, re-iniciar o desinstalar componentes en tiempo de ejecución sin necesidad de re-iniciar el sistema. Así, cuando MoRE detecta algún cambio en el contexto de ejecución, lo traduce en activaciones y desactivaciones de estos componentes permitiendo la reconfiguración del modelo del proceso en tiempo de ejecución [17]. Análogamente, tomando esta idea de activaciones y desactivaciones, si cada tarea del modelo del proceso de negocio (ya esté definida en el Modelo Base o en el Modelo de Variación) es implementada y ejecutada como un componente OSGi, MoRE-BP puede llevar a cabo las reconfiguraciones del modelo del proceso de negocio en tiempo de ejecución.

Reconfiguración del modelo del proceso al contexto de ejecución: Específicamente, para llevar a cabo las reconfiguraciones, MoRE-BP sigue las cuatro fases del modelo de referencia de IBM para bucles de control autonómico, también conocido como *bucle MAPE-K* [5]. Estas fases son: *monitorizar*, *analizar*, *planificar*, y *ejecutar* (ver figura 7).

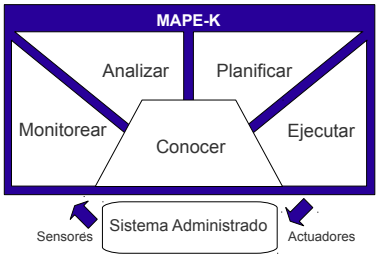


Fig. 7. Modelo de referencia para bucles de control autonómico de IBM

En nuestro caso, el *Sistema Administrado* corresponde al proceso de negocio que se está ejecutando, los *Sensores* capturan la información del contexto y los *Actuadores* llevan a cabo los cambios respectivos en la composición del proceso de negocio en tiempo de ejecución. A continuación se explican con detalle cada una de las fases del bucle:

- **Monitorizar:** En esta fase, se captura la información del contexto mediante el *Componente de Monitorización* de MoRE-BP. Este componente está compuesto por dos tipos de *Sensores*: 1) *Sensores de Cambios Manuales*, que monitorizan la información introducida por el usuario, y 2) *Sensores de Cambios Automáticos*, que monitorizan la información tomada de la infraestructura computacional. A continuación, la información de cambios contextuales que ha sido recibida por estos *Sensores* es insertada en el Modelo de Contexto definido anteriormente.
- **Analizar:** En esta fase, MoRE-BP revisa el Modelo de Contexto actualizado para saber si la información que contiene cumple alguna de las condiciones del contexto, definidas en el Modelo de Resolución, que determinan si el proceso debe cambiar. Por ejemplo, si el usuario ha introducido un país de destino que no requiere visado, entonces se cumple la condición "país_destino_requiereVisado = FALSO" que determina el *replacement* concreto que resuelve el *placement Tramitar visado* (ver figura 4).
- **Planificar:** En esta fase, MoRE-BP configura un *plan de reconfiguración* para el modelo. Este plan contiene el grupo de acciones de reconfiguración que deben llevarse a cabo para adaptar el modelo del proceso al nuevo contexto de ejecución. Así, en el caso de estudio presentado, dada la condición del contexto "país_destino_requiereVisado = FALSO" se encuentra la resolución *Omitir visado* que dice que el fragmento aplicable es el *replacement Omitir el Visado*. Concretamente, el *plan de reconfiguración* generado para llevar a cabo la resolución *Omitir visado* es el siguiente:

Ejecutar de acuerdo a la Resolución Omitir visado::
(-) Suprimir cualquier otro replacement en el placement Tramitar visado

(+) Colocar el replacement Omitir visado en el placement Tramitar visado

- **Ejecutar:** En esta fase, MoRE-BP actualiza el proceso en tiempo de ejecución de acuerdo al plan de reconfiguración generado en la fase anterior. Específicamente, MoRE-BP realiza las siguientes acciones de reconfiguración:
  - Acciones a nivel de Componentes: Cuando un *replacement* debe ser instanciado en un *placement* específico, MoRE-BP debe (1) parar los componentes OSGi de ese *placement* asociados al anterior *replacement* instanciado y (2) iniciar los componentes asociados al *replacement* que debe ser ahora instanciado. Todos los componentes se encuentran en un repositorio de lista de espera listos para ser instanciados.
  - Acciones de Flujo de Datos: Una vez se ha realizado la instanciación, los nuevos componentes deben conectarse al resto de componentes que componen la ejecución del proceso. Esta conexión se realiza mediante flujos de datos que son implementados utilizando la OSGi Wire Class.
  - Acciones en el Modelo: Después de que los componentes han sido conectados entre sí, MoRE-BP actualiza el modelo del proceso de negocio para que se ajuste a la nueva ejecución. MoRE-BP realiza esta actualización por medio de una reflexión parcial del proceso de negocio usando la introspección a su modelo. Esta operación se realiza mediante el Eclipse Modeling Framework [16], que permite consultar dinámicamente la estructura de modelos en tiempo de ejecución. Las acciones a realizar sobre el modelo se describen en mayor detalle en la siguiente fase.

  Nuestro prototipo soporta adaptaciones del proceso de negocio a nivel del *esquema del proceso* [18], que en nuestro caso, está representado con un modelo BPMN. Específicamente, las adaptaciones que se aplican al *esquema del proceso* se propagan a todas las instancias del proceso que se ejecutan con este esquema. Para asegurar que la adaptación se lleva a cabo de una forma correcta y consistente, las instancias del proceso en ejecución permanecen asociadas a la versión del esquema que fue utilizada antes de la adaptación, mientras que nuevas instancias se crean en base a la nueva versión del esquema. Esta aproximación apoya la co-existencia de instancias del proceso que están asociadas a diferentes versiones del esquema [18].
- **Conocer:** Nuestra propuesta hace un uso intensivo del *conocimiento* presente en los modelos definidos. Específicamente, los eventos del contexto están representados mediante ontologías en OWL, y la variabilidad se captura mediante *CVL*. La reconfiguración del modelo del proceso de negocio se realiza mediante la extracción de información de estos modelos. Para este fin, MoRE-BP utiliza las siguientes tecnologías de consulta a modelos:
  - El *SPARQL Protocol and RDF Query Language* (SPARQL) es la recomendación de lenguajes de consulta para tripletas RDF del W3C. SPARQL permite realizar consultas sobre el modelo de la ontología expresado en OWL. En particular, se utiliza la consulta de tipo INSERT para insertar nuevas tripletas en el grafo RDF de la ontología que representa a nuestro modelo del contexto. Además, MoRE-BP evalúa

los valores del Modelo de Contexto para encontrar si alguna condición del contexto se ha cumplido. Para realizar esta operación, MoRE-BP utiliza la consulta de tipo ASK.
- Con el fin de proyectar una resolución a la arquitectura del sistema, MoRE-BP realiza consultas al modelo de negocio mediante el *EMF Model Query framework* (EMFMQ). Básicamente, EMFMQ provee una API para construir y ejecutar declaraciones de consultas en una forma similar a SQL. Estas declaraciones pueden utilizarse para descubrir y modificar elementos del modelo.

4 Estado del arte

En el ámbito de la gestión de los procesos de negocio se han desarrollado distintas propuestas para facilitar la gestión de los cambios que pueden producirse en los procesos de negocio de acuerdo a cambios en el contexto. Estas propuestas proporcionan soporte a la variabilidad en tiempo de análisis y diseño y, en algunos casos, también en ejecución.

Para manejar la variabilidad que ocurre en tiempo de análisis y diseño, se encuentra la propuesta C-EPC (*Configurable EPC*) [2], que proporciona soporte a la variabilidad en modelos de procesos de referencia[6]. En C-EPC, las múltiples variantes de un proceso se modelan integradas, mediante nodos configurables, en un único modelo. Así, para un contexto determinado, el modelador configura (selecciona) la variante del proceso que más se ajusta a ese contexto. El problema de esta propuesta surge cuando la casuística del proceso es realmente alta, ya que el tamaño de los modelos crece rápidamente convirtiéndose en artefactos muy difíciles de manejar [19]. Además, después de realizar la configuración de la variante, el proceso se ejecuta de forma fija y sin posibilidad de adaptarse a los cambios que pueden aparecer durante su ejecución.

Otra propuesta que proporciona soporte a la variabilidad en tiempo de análisis y diseño se desarrolló en el contexto del proyecto PESOA [3]. Este proyecto se orientó al desarrollo y personalización de software y a las familias de procesos. Con este fin, se introdujo el concepto *Variant-Rich Process Models*, basado en la idea de construir un único modelo anotado donde representar todas las variaciones del proceso de negocio. Para representar adecuadamente la variabilidad dentro del modelo, se introducen en el modelo un conjunto de anotaciones que permiten representar los diferentes comportamientos del proceso asociados a la variabilidad. Sin embargo, la semántica que hay detrás de estas anotaciones es bastante compleja, complicando el entendimiento del modelo. Además, al igual que C-EPC, una vez se ha construido el modelo, al desplegarlo sobre el motor de ejecución, se convierte en un elemento fijo sin posibilidad de variar.

Para gestionar la variabilidad que ocurre tanto en tiempo de diseño como en tiempo de ejecución, se encuentra Provop *(PROcess Variants by OPtions)*

[6] modelos de procesos de negocio que formalizan las actividades recomendadas que deben realizarse en ciertos dominios

que propone un enfoque operacional para dar soporte a la variabilidad en los procesos de negocio [4]. Parte de la idea de que una variante del proceso puede obtenerse ajustando (configurando) un modelo base a un contexto determinado. Así, el modelo base se define con la variante del proceso más común y el resto de variantes se obtienen aplicando un conjunto de operaciones de cambio (INSERT, DELETE, MOVE and MODIFY) que lo modifican. De esta manera, los modelos se vuelven mucho más simples y entendibles. Durante la fase de ejecución, el proceso se despliega con todas sus posibles alternativas, mediante el uso de ramas condicionales, sobre el motor de ejecución. La evaluación del contexto en el punto de variación determinará por qué rama debe continuar el proceso. Sin embargo, esta forma de gestionar la variabilidad en ejecución resulta compleja e inmanejable cuando el número de alternativas del proceso es muy elevado.

5 Conclusiones y trabajo futuro

Este trabajo describe una propuesta para la gestión de procesos de negocio auto-adaptables en función de su entorno de ejecución. Para ello, se modela el proceso de negocio en cuatro modelos (Modelo Base, Modelo de Variación, Modelo de Resolución y Modelo de Contexto) previniendo las distintas alternativas (variantes) que el proceso puede tener (en función del contexto de ejecución). El proceso se ejecuta sobre el motor MoRE-BP mediante el uso de modelos en tiempo de ejecución. Cuando el motor detecta un cambio en el contexto que hace variar el proceso, siguiendo los modelos definidos en la primera fase, el proceso puede adaptarse automáticamente al nuevo entorno.

Aunque la utilización de OSGi para la implementación del prototipo es adecuada, en trabajos futuros estamos interesados en investigar cómo MoRE-BP podría modificar el proceso de negocio cuando éste se define en términos de un lenguaje de ejecución de procesos de negocio como pueda ser BPMN 2.0 o BPEL. También es conveniente investigar en un futuro cómo el proceso puede adaptarse a cambios inesperados que puedan surgir durante la ejecución del mismo. Estos cambios pueden conllevar la definición de nuevas alternativas que no fueron consideradas durante la fase de análisis y diseño del proceso. Por lo tanto, será necesario conocer los efectos que pueden tener estos cambios sobre el proceso y, también, cómo se comportará el proceso frente a ellos. Además, otro aspecto importante es el desarrollo de una guía metodológica completa y detallada donde se asista los modeladores en cómo poner en práctica, paso a paso, la propuesta aquí presentada: buenas prácticas de modelado, granularidades recomendadas, etc.

La propuesta presentada en este trabajo se desarrolla en el contexto de la administración pública, en particular en la Consellería de Infraestructura y Transporte de la Comunidad Valenciana. En esta Consellería se requiere de mecanismos para la definición y ejecución de la variabilidad en los procesos administrativos que allí se definen. Así, la Consellería constituye el contexto industrial en el que pretendemos llevar a cabo la validación de nuestra propuesta.

Referencias

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Number 978-3-540-73521-2. Springer-Verlag Berlin Heidelberg 2007 (2007)
2. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1) (2007) 1–23
3. Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: Variability mechanisms for process models. Technical report, BMBF-Project (2006)
4. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08). (June 2008) 154–161
5. IBM: An architectural blueprint for autonomic computing. Technical report, IBM (2006)
6. Haugen, O., Moller-Pedersen, B., Oldevik, J., Olsen, G., Svendsen, A.: Adding standardized variability to domain specific languages. In: Software Product Line Conference, 2008. SPLC '08. 12th International. (2008) 139 –148
7. Bayer, J., Gerard, S., Haugen, Ø., Mansell, J.X., Møller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J.P., Widen, T.: Consolidated product line variability modeling. In: Software Product Lines. (2006) 195–241
8. CVL-RFP: `http://www.omg.org/cgi-bin/doc?ad/2009-12-3`
9. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: The provop approach. Software Process: Improvement and Practice (2010)
10. Alena Hallerbach, Thomas Bauer, M.R.: International Handbook on Business Process Management. In: Configuration and Management of Process Variants. Springer (2010)
11. Hitzler, P., Sebastian, R., Krötzsch, M.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC, London (2009)
12. Dean, M., Schreiber, G.: Owl web ontology language reference. `http://http://www.w3.org/TR/owl-ref/` (February 2004)
13. Stanford: Protégé. `http://protege.stanford.edu/`
14. Grenon, P.: Defining extensions to wsmo for service oriented architectures for all capturing contextual information. Technical Report D3.4.7, Service Oriented Architectures for All (2009)
15. Cetina, C., Giner, P., Fons, J., Pelechano, V.: Autonomic computing through reuse of variability models at runtime: The case of smart homes. Computer **42**(10) (2009) 37–43
16. Eclipse: Eclipse modeling framework project (emf). `http://http://www.eclipse.org/modeling/emf/`
17. Cetina, C., Haugen, O., Zhang, X., Fleurey, F., Pelechano, V.: Strategies for variability transformation at run-time. In: Proceedings of the 13th International Software Product Line Conference. SPLC '09, Pittsburgh, PA, USA, Carnegie Mellon University (2009) 61–70
18. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66** (September 2008) 438–466
19. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information & Software Technology **52**(2) (2010) 127–136

Fault-Tolerant Business Processes *

Sanka Samaranayake, Ricardo Jiménez-Peris, Marta Patiño-Martínez

Facultad de Informática
Universidad Politécnica de Madrid, Spain
{ssamaranayake,rjimenez,mpatino}@fi.upm.es

Abstract. Service-oriented computing (SOC) paradigm promotes the idea of assembling application components into a network of loosely coupled services. Web services are the most promising SOC-based technology. A BPEL process definition represents a composite service that encapsulates some complex business logic including the invocation to other (external) web services. The complexity of a BPEL process together with the invocation of external services subject to network and computer failures requires countermeasures to tolerate this kind of failures. In this paper we present an overview of FT-BPEL, a fault-tolerant implementation of BPEL that copes both with failures of the machine running the BPEL process and network failures in a transparent way, that is, after a failure the system is able to resume the BPEL process consistently.

1 Introduction

Service-oriented computing paradigm promotes the idea of assembling application components into a network of loosely coupled services. Services are autonomous, independent entities that can be described, published, discovered and loosely coupled. Application developers can compose rapid, low cost and evolvable applications by discovering and invoking the network-available services to accomplish some task ranging from a simple query to execution of sophisticated business logic [11].
Composite services, services that invoke other services, often are long running activities, loosely coupled and span across multiple layers of service consumer and providers [12]. Web services are the most promising SOC-based technology. They use Internet as the communication medium and open Internet-based standards, including Simple Object Access Protocol (SOAP) [6] for packaging/routing data, Web Services Definition Language (WSDL) [7] for defining services, Business Process Execution Language for Web Services (BPEL-WS) [3] for orchestrating services. A BPEL process definition represents a composite service that encapsulates some complex business logic. Web services consists of complementary standards set, atop the core to provide various Quality-of-services including WS-Addressing [5] to encapsulate message routing information, WS-Reliable Messaging [4] for ensuring reliable message delivery etc. Due to the complexity and

* This work has been partially funded by the Spanish Research Council (MiCCIN) under TIN2010-19077, by the Madrid Research Foundation, S2009/TIC-1692 (co-funded by FEDER & FSE), and CCG10-UPM/TIC-5855.

long running nature of composite services they are susceptible to a wide variety of failures, for instance, lost messages due to unreliable communication links, crash of the services, crash of the composite service.
Inconsistencies may arise if the solution to tolerate computer failures (crash failures) is simply restarting a composite service. The process may invoke third party stateful services, for instance booking a hotel room or buying an airplane ticket. Repeating the process execution from the very beginning leads in such scenarios to undesirable situations where invocations are repeated. That is, there is no guarantee that a service invocation is performed *exactly once*. This paper presents the architecture and design of FT-BPEL, a framework for reliable web services orchestration built in BPEL. The framework implements passive replication, that is, the process state is persisted and in case of a crash, the process is restarted on a different machine from the point where the failure happened avoiding repeating the execution of the actions the process successfully executed before the failure. The framework tolerates communication failures.

The rest of the paper is organized as follows. Section 2 presents some concepts of fault-tolerance and design goals. Section 3 presents the proposed architecture. Section 4 describes how we deal with failures. Finally, Sections 5 and 6 present related work and conclusions.

2 Fault Tolerance

FT-BPEL deals with two kinds of failures: crash of the machine running BPEL and communication failures. The goal is to provide failure transparency, that is, after a failure the system recovers and resumes execution from the point where the failure happened avoiding duplicate requests. This is important for any stateful service. In order to tolerate crash failures, FT-BPEL implements the so called *passive replication model*. That is, the state of running processes is persisted during normal execution (*checkpoint*). If there is a failure, the process is started on a different computer and its execution resumes from the last checkpoint. The main challenge here is to avoid repeating the execution of the process from this point till the point where the failure happened, especially those actions that may change the state of the process or invoke another service. For instance, let us assume that a checkpoint of a BPEL process is made and then, it invokes an external service to buy a theatre ticket. The BPEL process fails before the response is received. When the process is resumed from the last checkpoint, the service invocation is repeated and the result is that two tickets are bought. These duplicate invocations are avoided by FT-BELP.

A similar situation arises in the presence of network failures. If the external service is executing on a different organization, network disconnections may happen and invocations may not reach its destination. So, the client of this service upon a failure notification, it may retry the invocation. It may happen that the invocation reached the service but the response did not reach the client because a disconnection happened in between. In this case, the client invocation will be executed twice.

The goal of FT-BPEL is to deal with these situations where a failure may lead to duplicate executions. This is an important issue in the context of web services where service autonomy must be preserved. We have designed our framework to meet the following goals:
Failure Transparency. FT-BPEL is able to recover from crash failures of the machine running BPEL processes in such a way that is completely transparent to the users and external services of the system.
Interoperability. FT-BPEL framework relies only on open Internet-based standards for web services including WS- Addressing for message correlation, WS-Reliable Messaging for reliable message delivery and duplicate filtering.
Ease of Use. FT-BPEL uses Apache ODE [1] as the orchestration engine and WSO2 Mercury [2] as the WS-Reliable Messaging provider. Any standard BPEL process definition can be deployed in FT-BPEL without any modification.

3 FT-BPEL Architecture

Figure 1 provides an architectural overview of FT-BPEL framework. Its execution environment consists of an ODE runtime as the BPEL orchestration engine, Mercury runtime as the WS-Reliable Messaging provider and FT-BPEL integration layer that mediates the communication between the orchestration engine and reliable messaging service provider.

WS-ReliableMessaging allows reliably sending SOAP messages between two web services over an unreliable infrastructure. It provides the following guarantees:*AtLeastOnce*, *AtMostOnce*, *ExactlyOnce* and *InOrder*. *AtLeastOnce* means that messages may be delivered more than once (duplicates). If a message cannot be delivered, an error is raised. *AtMostOnce* means that there will be no duplicates but a message may not be delivered. *ExactlyOnce* guarantees that a message is delivered exactly once (no duplicates). If a message cannot be delivered, an error is raised. *InOrder* guarantees FIFO delivery of messages. This property can be combined with the previous ones.

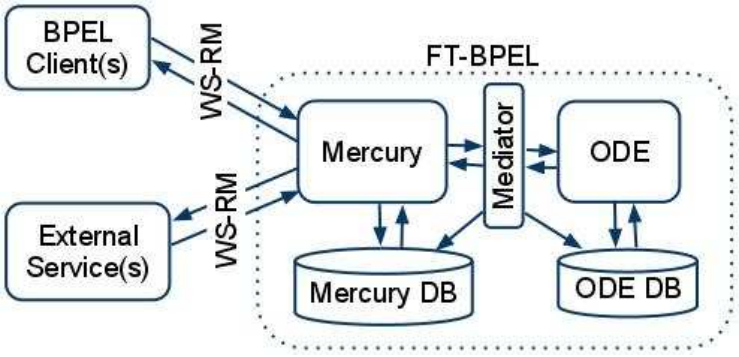


Fig. 1. FT-BPEL Architectural Overview

Our architecture uses *WS-ReliableMessaging* with *ExactlyOnce* delivery for the communication among web services. Both clients invoking a BPEL process and invocations from a BPEL process use *WS-ReliableMessaging* with Exactly-Once delivery guarantees.

WS-Reliable Messaging is transparent to the existing applications. In WS-Reliable messaging systems, there are handlers or agents which reside in the client's or server's SOAP processing engines that transfer messages, handle retries and deliver to the application. These agents are not visible at application level and they ensure that messages get retransmitted if lost or undelivered.

FT-BPEL mediates the communication between the orchestration engine and the reliable messaging service. FT-BPEL assumes that all clients and external services use a WS-Reliable Messaging protocol for reliable communication. Each message exchanged is passed through a logical reliable messaging channel called sequence. Each sequence carries a unique sequence identifier. We are using Mercury as the WS-Reliable Messaging provider. Mercury uses a database to store the state of the communication and guarantee the delivery of messages in case of failures. Mercury is able of retransmitting any messages that are outstanding in the outgoing buffer and the ones that are sent but not acknowledged after predefined (configurable) amount of time. It discards any duplicates. Therefore, any message losses due to network failures are dealt accordingly.

ODE also uses its own database to persist the state of each process instance after an activity is executed. So, if there is a failure, the instances running at the time the failure happened are restored and execution resumes from the last activity executed.

In order to provide high availability, the databases may be replicated in different nodes. We are currently using MySQL replication as replicated database. However, our implementation is independent of the database management system. High available LAN configuration can be used to detect a failure of FT-BPEL runtime and redirect any incoming traffic to FT-BPEL slave runtime.

4 System Failures and Recovery

In this section we present how we deal with failures during the execution of a request from a BPEL client to the BPEL server. Request from the BPEL server to other services are implemented in a similar way.

Upon the successful reception of an invocation request from a BPEL client via Mercury, FT-BPEL mediator will inject it to ODE runtime. If a failure occurs during communication between BPEL Client and FT-BPEL Server, Mercury will resume any incomplete reliable messaging sequences upon recovery. Mercury delivers any messages it receives to FT-BPEL mediator by executing a single callback function. After the completion of this callback, Mercury marks the sequence as a successful delivery by writing the appropriate changes to its database. FT-BPEL injects any invocation request to ODE runtime in the context of a transaction. If a failure occurs during this transaction, ODE runtime would be restored to state before the injection. Since this failure happens during

the callback from Mercury to FT-BPEL and while awaiting its completion, Mercury will re-attempt the delivery. Hence FT-BPEL recovers from these failures.

However, if the failure occurs just after the completion of the ODE transaction, but before Mercury marks the sequence as a successful delivery, duplicate invocations can be produced. Upon recovery, Mercury will re-attempt the delivery and the invocation request will get injected again to ODE. This leads to duplicate invocations of the BPEL process. FT-BPEL utilizes ODE's capability to associate an identifier (i.e. client key) with the request to be injected. ODE guarantees that the response will contain the same identifier. FT-BPEL mediator uses the reliable messaging sequence identifier through which it receives the request as the client key. Before FT-BPEL mediator injects the request to ODE runtime, it first checks for the existing of the client key in the ODE database. If it exists, the request is discarded as a duplicate.

When the response is ready, ODE runtime notifies FT-BPEL mediator by executing a callback function. FT-BPEL mediator extracts the client-key from the ODE response which is the sequence identifier from which it received the request. It writes the response message to the correct reliable messaging sequence in the Mercury database. This triggers Mercury to deliver the response to the BPEL client. This happens within a Mercury transaction. Upon completion, ODE runtime is notified explicitly. If a failure occurs during the Mercury transaction, Mercury runtime will be restored to the state before the response was written to its database. ODE runtime will re-execute the callback since it was not explicitly notified about the completion. Hence, the response will be delivered at the client.

However, if the failure occurs just after the completion of the transaction but before ODE is notified, the failure causes the ODE runtime to re-execute the callback which in turn will write the same response message to the reliable messaging sequence in Mercury database causing it to deliver the same response message to the client again. To prevent this, FT-BPEL mediator always checks whether the response exists in Mercury database before writing it. If exits, FT-BPEL mediator discards the response and notifies ODE runtime explicitly.

5 Related Work

Several techniques have been proposed for reliable service orchestrations in unreliable environments. [9] suggested the use of atomic execution of services and exception handling. [13] proposed the use of declarative, reusable fault handling logic (exception handling policies) and use of system infrastructure to generate an exception-aware process schema. [10] presented an approach where a BPEL process is annotated with recovery actions and then converting the annotated WS-BPEL into a standard BPEL. [8] presented a set of extensible recovery policies to specify how to handle and recover from typical faults in web services composition. Their approach delegated the enforcement of recovery policies to the underlying messaging middleware. In general, they adopt two fault-tolerant mechanisms, transactions for backward recovery and exception handling for for-

ward recovery. Fault recovery logic is embedded into process description and converted into standard BPEL process with various degrees of automation.
Our work is different as we focus on providing a fault-tolerant mechanism against system failures (crash failures) and not against faults that are thrown by services.

6 Conclusions

In this paper we have presented a fault tolerant framework for web services orchestration. We argue that state persistence combined with web services reliable messaging can be used for consistent crash recovery and guarantee 'exactly once' external service invocations. The framework is constructed using state of the art technologies. Currently, we are running a thorough evaluation of the system.

References

1. Apache ODE. http://ode.apache.org/.
2. WSO2 Mercury. http://wso2.org/projects/commons/mercury.
3. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, et al. Business process execution language for web services, version 1.1. *Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation*, 2003.
4. R. Bilorusets, D. Box, L.F. Cabrera, D. Davis, D. Ferguson, C. Ferris, T. Freund, M.A. Hondo, J. Ibbotson, L. Jin, et al. Web services reliable messaging protocol (WS-ReliableMessaging). *Specification, BEA, IBM, Microsoft and TIBCO, http://www-128. ibm. com/developerworks/library/specification/ws-rm*, 2005.
5. D. Box, F. Curbera, et al. Web services addressing (WS-Addressing), 2004.
6. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1, 2000.
7. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1, 2001.
8. A. Erradi, P. Maheshwari, and V. Tosic. Recovery policies for enhancing web services reliability. 2006.
9. A. Liu, L. Huang, Q. Li, and M. Xiao. Fault-tolerant orchestration of transactional Web services. *Web Information Systems–WISE 2006*, pages 90–101, 2006.
10. S. Modafferi and E. Conforti. Methods for enabling recovery actions in WS-BPEL. *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, pages 219–236, 2006.
11. M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.
12. C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, October 2003.
13. Liangzhao Zeng, Hui Lei, Jun jang Jeng, Jen-Yao Chung, and B. Benatallah. Policy-driven exception-management for composite web services. *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on*, pages 355–363, July 2005.

Mixing RASCI Matrices and BPMN Together for Responsibility Management*

Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés

Universidad de Sevilla, Spain
{cristinacabanillas,resinas,aruiz}@us.es

Abstract. Organizations need to manage the responsibility of the employees with respect to all the activities that are carried out daily. RACI matrices are used to this end, providing information about who must do what, i.e., the responsibility of each human resource regarding each activity, e.g. responsible for its execution, responsible for approving it once executed, etcetera. On the other hand, nowadays organizations increasingly use modelling notations to represent their processes, being BPMN the standard for business process modelling. In this paper we focus on a concrete type of RACI matrices called RASCI and introduce a novel approach to build RASCI-aware business process models in BPMN based on what we have called *RASCI patterns*. Furthermore, we explain how the transformation of information between RASCI matrices and RASCI-aware BPMN models can be (semi-)automatically performed. We believe the transformation from RASCI to BPMN and vice versa may be very useful to organizations, since it releases them from having to manually maintain BPMN models and RASCI matrices separately and it allows them to focus on only one of them for responsibility management.

Key words: RACI matrix, BPMN, RASCI pattern, RASCI-related BPMN activity, responsibility management

1 Introduction

Organizations need to manage the assignment of responsibilities of their members with respect to the activities that must be carried out within them. This means that not only associating functions to each member of the organization is necessary in order to have an action plan of the work performed by every member for every activity, but also providing a global view, that is, a way to display and organize these responsibility assignments, is required. This can be done by means of a Responsibility Assignment Matrix (RAM), also known as RACI matrix or Linear Responsibility Chart (LRC). This kind of matrices provide a way to plan, organize and coordinate work and consist of representing

* This work has been partially supported by the European Commission (FEDER), Spanish Government under the CICYT project SETI (TIN2009-07366); and projects THEOS (TIC-5906) and ISABEL (P07-TIC-2533) funded by the Andalusian Local Government.

certain associations for each activity, such as who is in charge of performing the activity and who must be informed when the activity is done [1]. Several variants extending the functions considered in traditional RACI matrices have appeared (e.g., RASCI matrices).

Besides, organizations need to manage their business processes somehow in order to organize the execution order of the activities carried out in the company, both to complete internal work and to provide services to external users. We remind the reader that a business process represents the execution flow of the activities necessary to complete certain procedure within one or more organizations.

The joining point between RACI matrices and business processes are the activities. The activities to which RACI matrices refer may be actually activities included in business processes. Specifically, we could think of building one RACI matrix for each business process model. Business processes can be modelled in different ways, using different modelling languages. Business Process Modelling Notation (BPMN) is the de-facto standard for this purpose [2]. Its current version includes improved mechanisms to deal with data objects and resources, but there is not yet an easy-to-use standardized way to express and manage resource assignments in the activities of the processes. This makes it harder to join the information included in RACI matrices with the activities represented in business process models, which would be necessary in order to make it easier the management of human responsibilities together with the business process activities performed in an organization. Having RACI matrices and BPMN models separately has several drawbacks:

- An organization usually tracks the execution of business processes. However, *complete* information about resource assignments is still missing in business process models, so there is not a way of knowing all the responsibility assignments without taking a look at the RACI matrix.
- Keeping the information of both elements consistent is difficult, since the RACI matrix must be updated manually when the business process is modified.
- The RACI matrix does not understand about organizational models and, thus, inconsistencies between the responsibilities assigned to the activities and the hierarchical structure of the organization could come up. For instance, the RACI matrix could capture a situation in which an employee delegates work to a boss, which is unlikely to happen in an organization.

Therefore, coming up with a way to synchronize all the elements is necessary. The main goal of this work is to integrate a specific type of RACI matrices, *RASCI matrices*, with BPMN, with the aim of easing responsibility assignment management to organizations that use BPMN to model their business processes. Integrating a RASCI matrix into a BPMN model means enriching the process model with RASCI information, i.e., making the model RASCI-aware. In this paper we explain how this is possible manually or semi-automatically, and how we can automatically update a RASCI-aware BPMN model from changes applied

to the RASCI matrix related to the corresponding business process. To do so we use the extension capabilities provided by BPMN 2.0 to create new types of activities (referred in this paper as *RASCI-related BPMN activities*), and we introduce a set of *RASCI patterns* that make it easier the transformation from RASCI to BPMN.

On the other hand, the inverse procedure is also interesting, i.e., given a RASCI-aware BPMN model we could think of automating the generation and maintenance of the associated RASCI matrix. We describe how this can be done automatically.

The main advantages the RASCI patterns, the BPMN extension and the transformations from RASCI to BPMN and vice versa introduced in this paper provide to organizations are the following:

- The transformations of information between RASCI matrices and BPMN models allow focusing on only one view of a business process (being it the execution flow shown in the models or the responsibility assignments included in the matrices) instead of manually maintaining both views separately. We believe it may imply important effort saving to organizations.
- As the proposed solution to insert RASCI information in BPMN is based on BPMN semantics, the resulting RASCI-aware business process models are BPMN-compliant, so they can be run in any BPMN engine.

In section 2 we describe RASCI matrices and introduce a scenario that will be used as use case in the rest of the paper. Section 3 introduces aforementioned RASCI patterns and RASCI-related BPMN activities. In Section 4 the procedures to generate and maintain RASCI matrices from BPMN models and vice versa are described. Brief related work can be found in Section 5. Finally, conclusions drawn from this work and future work on this field are presented in Section 6.

2 RASCI Matrices and BPMN. Use Case

RACI matrices are usually used to associate activities with resources (individuals or groups).The following functions, called *roles* in RACI[1], must be indicated for each activity performed in an organization:

- *Responsible (R):* person who must perform the work, responsible for the activity until the work is finished and approved by an accountable. There is typically one person responsible for an activity.
- *Accountable - also Approver or final Approving Authority - (A):* person who must approve the work performed by the person responsible for an activity, and who becomes responsible for it after approval. There must be one and only one accountable for each activity.

[1] We will use the term *RACI/RASCI role(s)* from now on in order to not mistake them for organizational roles.

- *Consulted - sometimes Counsel - (C):* this role includes all the people whose opinion is sought while carrying out the work, and with whom there is two-way communication.
- *Informed (I):* person who is kept up-to-date about the progress of an activity and the results of the work, and with whom there is just one-way communication. There may be more than one person informed of the work of an activity.

Using these four RACI roles we can build a matrix where rows represent activities, columns are (human) resources and each cell contains zero or more RACI initials indicating the degree of responsibility of *such* resource on *such* activity. The definition of resources can be done at different levels: (i) although it is not very convenient, small companies may use individual resources (persons) in each column; (ii) in most cases, columns are likely to represent positions or roles played inside an organization; (iii) at a very high level we could find RACI matrices in which each column would refer to, for instance, a work group or organizational unit.

There are several variants of the original RACI matrix. Some are based on extending the number of RACI roles to be considered for every activity, e.g, RASCI or RACI-VS. Others give different meaning to the RACI initials. In this paper we will build on RASCI matrices (cf. example in Table 1) because they use a function that may be interesting specially to IT organizations, where work can usually be delegated to other people in order to complete certain activity.

- *S (Support):* people who may assist in completing an activity, i.e., the person in charge can delegate work to them. Unlike *Consulted*, who may provide input to the activity (i.e., information helpful to perform some work), *Support* will contribute in the completion of the activity.

As aforementioned in this paper, we could think of having one RASCI matrix for each business process run in an organization. The matrix would thus contain information about every activity carried out in a business process and the organizational roles involved. Figure 1 shows a BPMN diagram that represents a *collaboration* in which two business processes participate: one business process at pool *Research Vice-chancellorship* and another one at pool *ISA Group*. We remind that in BPMN a process takes place in a single pool. Diagrams with two or more pools, in which messages between the pools are exchanged, are called collaborations[2]. The collaboration in the figure illustrates a simplified version of the process to manage the trip to a conference (according to the rules of the University of Seville), from the submission of the final version of an accepted paper to the booking of the transport tickets and the accommodation. The collaboration starts with the submission of the Camera Ready version of a paper, and it continues when one of the authors fills up a form requesting for authorization both to travel to the venue place and to take the funds from some funding source. This authorization must be signed by some person in charge of account

[2] We are using concepts from BPMN 2.0 throughout the paper [2].

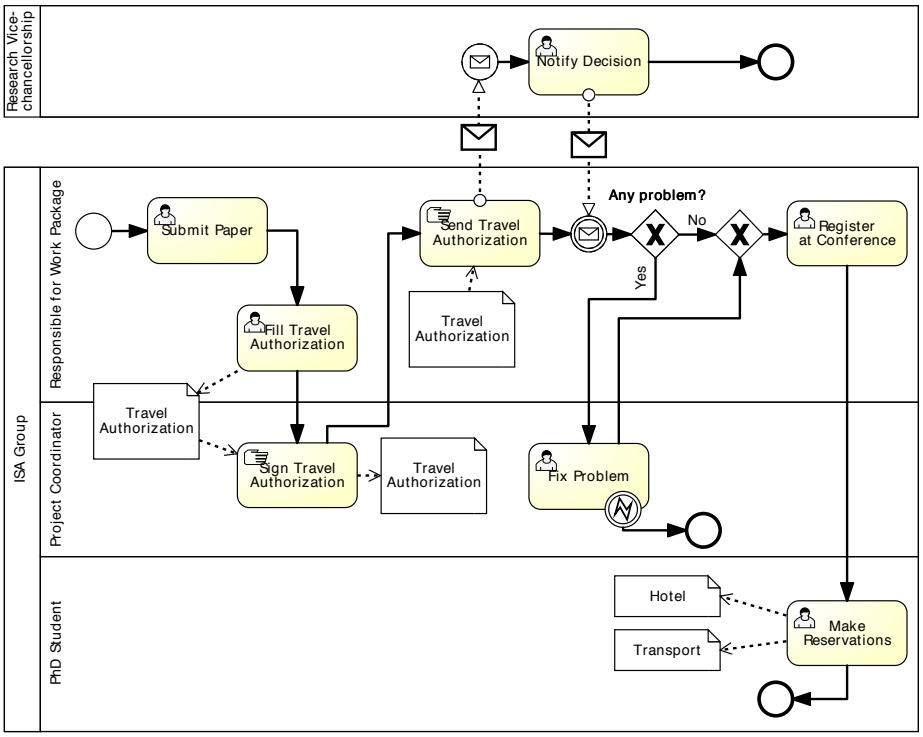


Fig. 1: Conference Travel Management Process

management related to the applicant. The travel authorization is then sent for revision to entity *Research Vice-chancellorship*, where someone might approve and sign the document. The approval decision is communicated to the applicant. If it is favourable, then the process continues with the registration to the conference and the booking of the proper tickets. Otherwise, corrective actions must be carried out in order to fix the problems with the travel authorization, and the process will go on until the reservations are made, as long as nothing wrong happened when fixing the problems.

We will focus on the business process carried out at pool *ISA Group*. Imagine that process takes place in an organization whose organizational model is the one in Figure 2. This model corresponds to an organization that uses the concepts Person, Position, Role and Organizational Unit according to an organizational metamodel described by Russell et al. [3]. Specifically, the model in the figure contains six positions (Project Coordinator, Account Delegate, Senior Technician, Administrative Assistant, Responsible for Work Package and PhD Student) that are members of one organizational unit (Project THEOS), and seven persons occupying these positions. Given this structure, Table 1 could be the RASCI matrix implemented in the organization. We are using positions as resources in the matrix, but any other kind of resource could have been used.

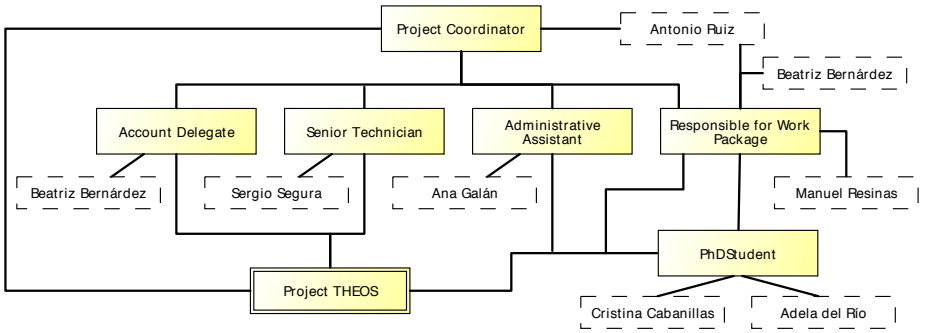


Fig. 2: Excerpt of the organizational model of *ISA Group* from a project perspective.

Table 1: RASCI matrix for the process at pool *ISA Group* of Figure 1

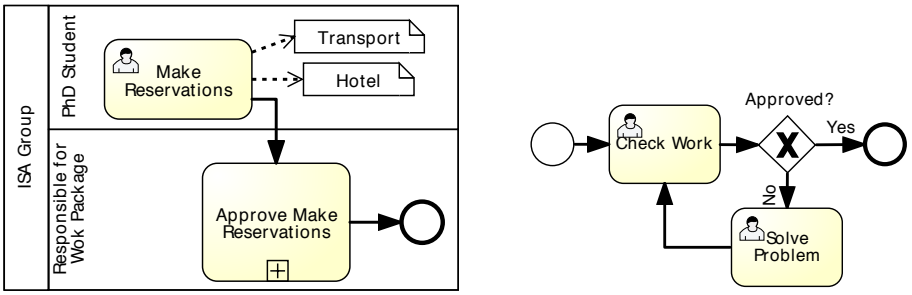
	Project Coordinator	Account Delegate	Senior Technician	Administrative Assistant	Responsible for Work Package	PhD Student
Submit Paper	I				R/A	S
Fill Travel Authorization	C	C		S	R/A	S/C
Sign Travel Authorization	R/A					
Send Travel Authorization				S	R/A	S
Fix Problem	S/C	S/C	C	S	R/A	S/C
Register at Conference	C/I				R/A	S
Make Reservations	I			S/C	A	R

This scenario will be used as use case in this paper. Another RASCI matrix based on the organizational model of the corresponding organization would be required for the process at pool *Research Vice-chancellorship*.

In the following we will describe some procedures that could be used to put RASCI matrices and BPMN models together, and some guidelines on how to take into account the organizational structure will be provided as well.

3 Introducing RASCI into BPMN. RASCI Patterns and RASCI-Related BPMN Activities

The first and main goal of this work is to find a way to include RASCI information in BPMN models. In the following we propose a set of RASCI patterns that represent the responsibility assignments used in RASCI for each RASCI role, in order to make business process models RASCI-aware. Firstly, we must find a way to represent resources in a BPMN model. We could use the pools and lanes provided by BPMN for this purpose, by giving them a semantics that



(a) Accountable Pattern

(b) Sub-process *Approve Make Reservations*

Fig. 3: Implementation of the Accountable Pattern. Example

now is missing [2]. We remind the reader that, according to BPMN definitions, a pool usually represents a participant in a business process, and may contain several lanes, which can be seen as participants at a lower level (e.g., a partner entity) or as a way of grouping and organizing participants. However, these two concepts are still a bit abstract and are not used in a standardized way.

Responsible Pattern. If we give the name of persons, positions, roles, or even organizational units to the lanes of a process and consider that the corresponding resource is responsible for all the activities within its lane, RASCI role Responsible (R) becomes implicitly represented in BPMN models. This way, *Responsible Pattern* consists of interpreting that the resource corresponding to the pool/lane in which an activity is placed is responsible for its execution. To support this interpretation, we must assume that there is only one resource responsible for the execution of each activity. In Figure 1 we are using this implementation and, hence, the model has three lanes representing the R's that appear in Table 1. We can consider that pools have the same meaning as lanes as long as they do not contain lanes within. Otherwise, they will be meaningless and the information will be provided by their lanes. Notice that the number of lanes in a pool might increase with respect to the traditional meaningless swimlanes.

Accountable Pattern. This pattern consists of an approval activity placed in the lane of the resource accountable for the activity in question. It can be seen (and modelled) as a sub-process. Only if R and A coincide this "extra" sub-process will not be necessary. We cannot generalize what to do if the approval process fails (i.e., the work is not approved), since it depends on the specific business process being modelled. The simplest case is to assume that either the activity is approved and the process goes on, or that otherwise the organization gets to solve the problem and the process can continue. Therefore, in our use case we should add at least an activity called *Approve Make Reservations* in the lane representing position *Responsible for Work Package* (cf. Figure 3).

Support Pattern. RASCI role S is played while carrying out the specific activity being represented. Therefore, the activity that requires support must be

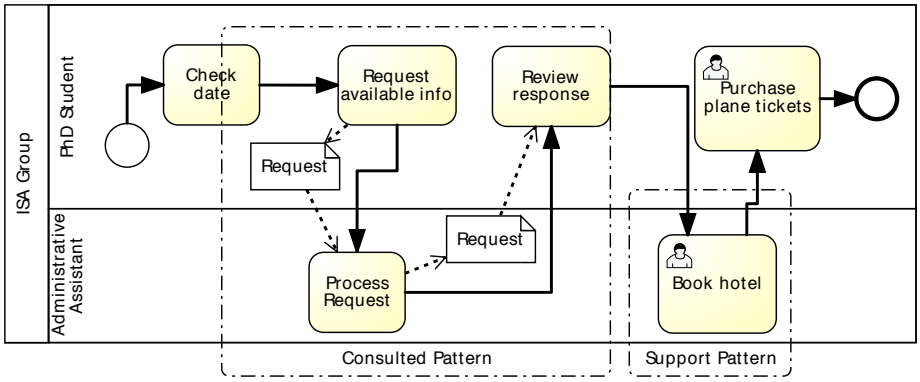


Fig. 4: RASCI patterns *Support* and *Consulted* in sub-process *Make Reservations*

modelled as a sub-process that contains the proper assignments to represent the delegation of work to other resources, which consists of tasks to perform in order to complete the work of the main activity (called *responsible sub-process* from now on in this paper). These tasks are associated to the corresponding resource (lane) in the sub-process, according to the RASCI matrix. Figure 4 shows the content of the now responsible sub-process *Make Reservations* and illustrates the use of *Support Pattern*.

Consulted Pattern. RASCI role C is also played inside the responsible sub-process and it can be seen as a request-response procedure in which intuitively no work is required. It could thus be modelled as messages exchanged between two resources, so one resource asks for something and another resource answers. However, in BPMN messages cannot be passed between lanes of a pool, so we would need at least two pools to model this pattern, making it no longer be a process but a collaboration. As we are associating RASCI matrices with single business processes, modelling a collaboration inside a responsible sub-process including resources of the main process in different pools would cause an inconsistency. Furthermore, the resource consulted may need some time to process the request and respond to it, so the request-response procedure is time-consuming and there is actually work to be done. To deal with this RASCI role we propose to use BPMN common tasks using the pattern shown in Figure 4. Notice that we make data objects play a role similar to messages' in collaborations in order to provide a mechanism to exchange information.

Informed Pattern. Unlike in RASCI role C, in RASCI role I no response is expected. We are focused on intra-organizational processes (related to a single organizational model) and, thus, the resource informed can be included as a participant of the process (if it was not yet) by means of a new lane. This resource receives the information but the normal execution flow of the process can continue in parallel. As defined in Section 2, people can be informed also about the progress of an activity, so we could include this pattern inside the responsible sub-process as well. *Informed Pattern* is shown in Figure 5, in which

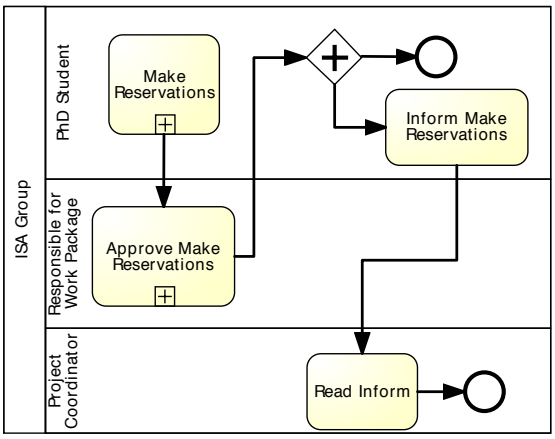


Fig. 5: *Informed Pattern* and summary of all patterns for activity *Make Reservations*

there is also a summary of all the patterns for activity *Make Reservations* of our use case. Note that in this case the process ends up after informing but, as aforementioned, in other scenarios the business process may continue.

3.1 Improving RASCI Patterns by Extending BPMN

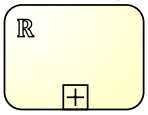
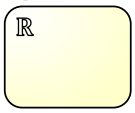
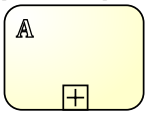
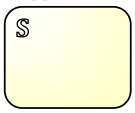
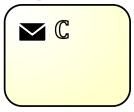
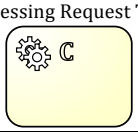
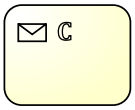
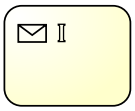
One of the problems related to the introduction of RASCI into BPMN models like explained above is that we can mistake RASCI structures for common BPMN structures, since they all use elements defined in BPMN 2.0 and the different interpretation comes from the meaning we have given to the structures used to model the RASCI patterns. This makes it difficult to recognize RASCI patterns and, hence, also the automatic extraction of RASCI information from a RASCI-aware business process model.

We propose the extension of BPMN with a set of RASCI tasks and collapsed sub-processes to help represent the RASCI patterns defined in Section 3. Table 2 contains such proposal, together with some constraints regarding where these activities can appear in a model and the naming conventions. Furthermore, the last column indicates whether the presence of the activity is necessary to be able to analyse a RASCI-aware BPMN model. Notice that it does not say whether the RASCI role is mandatory (cf. RASCI roles descriptions in Section 2), but whether the RASCI-related activity *must* appear in a RASCI-aware BPMN model in case the RASCI role exists in order to be able to identify relevant RASCI information.

4 Transformation of Responsibility-Related Information

The RASCI patterns and the RASCI-related BPMN activities provide the support necessary to include RASCI information in BPMN models maintaining the

Table 2: RASCI-related BPMN activities

RASCI Role	RASCI Sub-process/Task	Placement & Naming Constraints	Mandatory Icon?
Responsible	Responsible Sub-process  Responsible Task 	- Placed in any lane - Whatever name	No. The resource in whose lane the activity is placed is automatically responsible for the activity
Accountable	Approval Sub-process 	- Placed in any lane - Its name must contain the name of the referred activity	Yes
Support	Support Task 	- Always inside a responsible sub-process - Whatever name	No. Inside a responsible sub-process, any task in a lane representing a RASCI role (except R) may be interpreted as support
Consulted	Request Task 	- Always inside a responsible sub-process - In the same lane as the referred responsible sub-process - Whatever name	Yes
	Processing Request Task 	- Always inside a responsible sub-process - After the request task but in a different lane - Whatever name	If there is a request task
	Response Request Task 	- Always inside a responsible sub-process - After the processing request task and in the same lane as the request task - Whatever name	If there is a request task
Informed	Inform Task 	- In the same lane as the referred responsible sub-process/task - Its name must contain the name of the referred activity	Yes
	Read Inform Task 	- After the referred responsible sub-process/task but in a different lane - Its name must contain the name of the referred activity	If there is an inform task

BPMN 2.0 semantics, at the same time as they ease the transformation between both views of the responsibility assignment, as explained below.

4.1 Transformation from RASCI to BPMN

The basis to do this transformation has been set in Section 3. The automatic introduction of RASCI information in a BPMN model from scratch (i.e., in a BPMN model that knows nothing about RASCI) may not be possible, since it may require to modify the control flow of the business process and this could distort the real behaviour expected for the process. However, once a BPMN model is RASCI-aware, some changes[3] performed in the RASCI matrix could be automatically updated in the model without negative implications:

- *Changing the resource in charge of an activity.* It implies moving the responsible task/sub-process to a different lane of the process model according to the new resource responsible for the activity. If it is not a current participant in the process, the lane will be added. The corresponding activities approval sub-process and inform task, if existing, must be moved to the new lane as well in order to comply with the constraints described in Table 2. If the activity is a responsible sub-process, its content may also change.
- *Changing the resource accountable for an activity.* Like before, this change implies moving the approval sub-process to a different lane of the business process model according to the new accountable for the activity. If it is not a current participant in the process, the lane will be added.
- *Changing the support of an activity.* If any S is changed in the RASCI matrix, the support task(s) related to the activity for such resource must be moved to a different lane in the responsible sub-process. If it is not a current participant in the process, the lane will be added to such sub-process.
- *Changing the consulted of an activity.* Similarly to the previous case, this change implies moving the processing request task(s) to a different lane in the responsible sub-process. If it is not a current participant in the process, the lane will be added to such sub-process.
- *Changing the informed of an activity.* This change implies moving the read inform task(s) to a different lane in the main business process model and/or inside the responsible sub-process, adding new lanes if necessary.

Obviously, if any lane becomes empty after the update, it should be removed from the model. We assume the new RASCI matrix is consistent regarding the responsibility assignments, i.e., every resource in the matrix belongs to the organizational model and the matrix does not contain situations in which, for instance, the same resource is responsible for an activity and plays the RASCI role C too. This checking could be done directly from the RASCI matrix or while doing the transformation into BPMN (so detection of inconsistencies could be automated), but it is out of the scope of this paper.

[3] A change means assigning the responsibility to a different resource. The addition and removal of RASCI roles in the matrix could modify the process behaviour and, thus, these actions are not considered changes.

4.2 Transformation from BPMN to RASCI

Both generation and update of a RASCI matrix from a RASCI-aware BPMN model can be automated, since all the information required can be found in the process model and filling up a RASCI matrix given the necessary information is nothing complicated.

We are going to describe how to generate a RASCI matrix from a BPMN model, for which a RASCI analyser should be developed. Knowing that, applying the updates of the model to the RASCI matrix is straightforward.

- *Filling the name of rows and columns.* In principle, there will be a row for each non-RASCI activity and/or responsible task/sub-process in the BPMN model. We could opt for being stricter and consider only the responsible tasks/sub-processes (thus ignoring the rest). Columns will correspond with the lanes of the BPMN model and the lanes included in responsible sub-processes.
- *Obtaining the resource in charge of an activity.* An R will be added to the cell corresponding to the resource represented by the lane where the non-RASCI activity or responsible sub-process/task (depending on the degree of strictness) is placed in the model. We remind the reader that we are assuming there is only one resource responsible for each activity.
- *Obtaining the resource accountable for an activity.* An A will be added to the cell corresponding to the resource represented by the lane where the approval sub-process related to the activity is placed in the model. The absence of approval sub-process will be interpreted as if R and A were the same resource.
- *Obtaining the support of an activity.* An S will be added to the cells corresponding to the resources represented either by the lanes where the support tasks are placed in the responsible sub-process, or by any lane in such sub-process (different from the lane of RASCI role R) that contains tasks, according to the degree of strictness.
- *Obtaining the consulted of an activity.* Letter C will be added to each cell of the matrix corresponding to the resource represented by the lane where a processing request task is placed in the responsible sub-process.
- *Obtaining the informed of an activity.* An I will be added to the cells corresponding to the resources represented by the lanes where the corresponding read inform tasks are placed in the BPMN model.

5 Related Work

To the best of our knowledge there is not yet work aimed at mixing RASCI matrices (nor RACI matrices) and business processes together. Some information about how to interpret the information of RACI matrices can be found in [4].

However, there is some work related to the introduction of responsibility assignments in business process models. Meyer has worked on the extension of BPMN 1.1 to manage resource allocation in business process models [5]. In

the same direction, Cabanillas et al. have developed language RAL (Resource Assignment Language) to express resource assignments with the aim of using it as an extension of BPMN 2.0 [6]. The authors have managed to reason about resources by giving formal semantics to RAL. La Rosa et al. have presented a rich metamodel for capturing role-task and object-task associations embodied in the EPC notation, which can be transposed to other notations [7]. Nakatumba et al. have analysed and characterised resource behaviour after business process execution from event logs using process mining [8]. An optimal approach to allocate the most proficient set of employees for a business process from event logs based on Hidden Markov Models is introduced in [9]. Finally, Russell et al. have described a set of Workflow Resource Patterns aimed at capturing he way resources should be managed in workflows and have analysed the support provided by some workflow tools [3].

The cited work does not even mention RACI/RASCI as an alternative or as something to consider for responsibility assignment. Nevertheless, it is one of the methods used in organizations nowadays to manage their resources and its relation to business processes is quite close.

6 Conclusions and Future Work

Resource management and responsibility assignment in business processes is still a quite neglected research field. Little work dealing with this issue has been published so far and most of it is focused on specifying the resources in charge of executing the activities of a process, forgetting other important roles that are daily used in organizations. Here is where the idea of RACI matrices appears as a mechanism to represent the responsibility assignments referring not only to the execution of activities, but also to the approval of the work performed and the communication of the results to the convenient resource, among other roles.

In this paper we have mixed RASCI matrices and BPMN together, providing RASCI patterns to represent RASCI information in BPMN models and proposing an extension of BPMN 2.0 consisting of new types of activities. The goal was to obtain RASCI-aware business process models that could be analysed, so transformations between RASCI matrices and RASCI-aware BPMN models became easier and could even be done automatically.

From this work we can conclude that providing BPMN with mechanisms to express responsibility assignments maintaining the basic semantics of BPMN is possible and may be quite useful to avoid organizations having to handle business process models and RASCI matrices separately. We believe the notions given in this paper may save human effort to organizations and constitute an important first step towards introducing resource management into the standard notation for business process modelling. Although this approach has the advantages defined in Section 1, there are also some drawbacks to be taken into account:

– Much information must be added to the business process models to make them RASCI-aware, even for small ones. It implies bad scalability, since complex BPMN models may be unreadable after introducing RASCI patterns.

- Aggressive modifications in the RASCI matrix of an organization (e.g., addition or removal of RASCI roles in some activities) may involve important changes in the BPMN model, so special attention must be put on keeping the expected behaviour of the business process.
- The approach is limited to intra-organizational business processes with only one resource responsible for each activity (this is quite reasonable though).

We plan to study different alternatives to mix RASCI matrices and large BPMN models together, e.g. using the activity properties to insert RASCI information instead of including new activities in the model. It could solve problems one and two.

Future work in the line of this paper is the implementation of the extension of BPMN 2.0 with the activities proposed in Table 2 to enable modelling RASCI-aware BPMN models, and the implementation of all the procedures described in Section 4. Besides, this work should be extended to consider cross-organizational aspects and, hence, overcome the third drawback aforementioned.

Acknowledgements

We would like to thank Rafael Pastor for providing us information about how they work with RACI matrices at the Quality Office of the Information Technology Department of the Andalusian Health Service and for the discussion generated about this topic.

References

1. D. O. Conchúir, "Human resource management processes," in *Overview of the PMBOK Guide*, pp. 129–145, Springer Berlin Heidelberg, 2011.
2. "Bpmn 2.0," recommendation, OMG, 2011.
3. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond, "Workflow resource patterns: Identification, representation and tool support," in *CAiSE*, pp. 216–232, 2005.
4. M. Smith, "Role and responsibility charting (raci)," in *Project Management Forum (PMForum)*, p. 5, 2005.
5. A. Meyer, "Resource perspective in bpmn - extending bpmn to support resource management and planning," Master's thesis, Hasso Plattner Institute, Potsdam (Germany), 2009.
6. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Towards the definition and analysis of resource assignments in bpmn 2.0," tech. rep., Universidad de Sevilla, April 2011.
7. M. La Rosa, M. Dumas, A. H. ter Hofstede, J. Mendling, and F. Gottschalk, "Beyond control-flow: Extending business process configuration to resources and objects," tech. rep., 2007.
8. J. Nakatumba and W. M. P. van der Aalst, "Analyzing resource behavior using process mining," in *Business Process Management Workshops*, pp. 69–80, 2009.
9. H. Yang, C. Wang, Y. Liu, and J. Wang, "An optimal approach for workflow staff assignment based on hidden markov models," in *OTM Workshops*, pp. 24–26, 2008.

Modelado de Responsabilidades en Sistemas Basados en Servicios

José Raúl Romero[1], Juan Ignacio Jaén[2] and Antonio Vallecillo[2]

[1] Dept. Informática y Análisis Numérico. Universidad de Córdoba, España
[2] Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga, España
jrromero@uco.es, {jijaen,av}@lcc.uma.es

Abstract. Al crecer la complejidad de los sistemas basados en servicios se hace patente la necesidad de contar con lenguajes y herramientas que permitan validar los diseños y razonar sobre su comportamiento. Un tema poco contemplado hasta la fecha es el de la responsabilidad de los agentes que intervienen en este tipo de sistemas, así como la especificación de su comportamiento teniendo en cuenta los permisos y obligaciones asociados a cada agente y a cada acción. Este artículo presenta una primera propuesta para modelar sistemas basados en servicios que hace uso de conceptos de lógica deóntica, su reificación en términos de objetos del sistema, y el uso de una herramienta para la especificación y simulación de modelos específicos de dominio siguiendo las pautas de la Ingeniería dirigida por Modelos (MDE).

Keywords: SOA, responsabilidades, lógica deóntica, Ingeniería dirigida por Modelos, Simulación

1 Introducción

Desde hace varios años, la computación orientada a servicios viene tomando especial relevancia por los principios de diseño en los que se apoya y los beneficios que ofrece a nivel organizativo: interoperabilidad, agilidad y compromiso entre negocio e IT (*Information Technologies*, Tecnologías de la Información). En realidad, frente a paradigmas anteriores como la orientación a objetos (OO), o el desarrollo de software basado en componentes (DSBC), la orientación a servicios se enfrenta al reto de construir software en forma de servicios, esto es, piezas de software débilmente acopladas, altamente reutilizables y, fundamentalmente, centradas en aspectos concretos del negocio. Se trata, pues, de cambiar la visión que actualmente tienen muchas empresas de sus departamentos de desarrollo y favorecer su integración en el contexto organizativo y empresarial para el que trabajan, así como los productos que desarrollan. De esta forma, el esfuerzo requerido en la automatización de nuevos procesos de negocio se verá reducido si los proyectos de desarrollo IT se fundamentan en estas piezas personalizadas para las necesidades específicas de la empresa.

Diversas propuestas arquitectónicas ofrecen soluciones y mecanismos de especificación de estos productos (p.ej. SOA). Otros marcos arquitectónicos de

empresa (del anglicismo *Enterprise Architecture Framework*) también se adaptan a estos criterios de diseño y ofrecen soluciones para estructurar y modelar adecuadamente desde las necesidades y objetivos del sistema en su contexto organizativo hasta su infraestructura computacional (p.ej. RM-ODP [1]).

Una carencia que normalmente presentan estos métodos de especificación es la forma de modelar cómo los usuarios u otros agentes interactúan con el sistema desde el punto de vista del negocio. En general, esta problemática se aborda definiendo un conjunto preestablecido de restricciones explícitas sobre el comportamiento de los usuarios o acerca del modo en que agentes externos interactúan con el sistema. Se hace necesario, por tanto, hacer más explícito a nivel de negocio el contrato entre un sistema y sus usuarios (o participantes externos), de modo se le permita al arquitecto de negocio validar esta capacidad de interacción del sistema no sólo como una expresión de las operaciones que ofrece para ello, sino estableciendo contratos formales que expresen y definan el comportamiento adecuado para cualquier entidad externa que se considere usuaria del sistema. Obsérvese además la importancia que supone permitir al arquitecto que razone sobre el modelo realizado antes de abordar su desarrollo.

Un modo de abordar de forma precisa este tipo de especificaciones es mediante la lógica deóntica [2], una rama de la lógica que permite expresar el comportamiento del sistema en términos de permisos, obligaciones o prohibiciones. En este trabajo, proponemos un lenguaje específico de dominio (DSL, *Domain Specific Language*) que permita realizar especificaciones gráficas de nuestro sistema y sus participantes de forma que posteriormente se pueda razonar sobre ellas. Una vez modelados el negocio, los participantes externos y sus responsabilidades, debemos definir los mecanismos que permitan el razonamiento sobre esta especificación. Para ello, hacemos uso de la herramienta e-Motions [3], una herramienta de validación formal ejecutada sobre Maude [4], un lenguaje formal ejecutable basado en lógica de reescritura, que permite realizar validaciones formales utilizando DSL gráficos conformes a metamodelos que definen el dominio del sistema, y cuyo comportamiento viene determinado por reglas.

2 Modelado conceptual de responsabilidades

Como se expuso anteriormente, es necesario poder articular de manera clara restricciones no sólo sobre operaciones propias del sistema sino también sobre el comportamiento de cualquier participante externo. En este sentido, para abordar la especificación del comportamiento de un sistema, en primer lugar, se han de identificar todos los posibles estados que puede presentar para, posteriormente, en base a algún conjunto de reglas, normas o políticas, establecer cuáles de todos esos estados posibles identificados son más restrictivos que el resto.

Así pues, el comportamiento de un sistema está dirigido por una colección de restricciones que contemplan tres tipos de situaciones: *(a)* las que son posibles que ocurran y, por tanto, se han de permitir; *(b)* situaciones que no pueden ocurrir o prohibidas y *(c)* situaciones que tienen que ocurrir obligadamente. Todos estos conceptos (permisos, prohibiciones y obligaciones, respectivamente)

adoptados de la lógica deóntica estándar (SDL, *Standard Deontic Logic*) [2], no son independientes entre sí. Por ejemplo, se podría considerar una obligación continua en el tiempo como la prohibición de no hacer algo, una prohibición como el opuesto a un permiso, etcétera.

La lógica deóntica nos permite conocer si se han producido violaciones en alguna de las restricciones, aunque no se permite conocer sobre quién recae dicha responsabilidad exactamente, puesto que la responsabilidad se difumina en el conjunto global del sistema.

Anteriormente se han presentado también otras propuestas iniciales para el modelado de responsabilidades en sistemas basados en servicios. Por ejemplo, en [5] los autores proponían un metamodelo de responsabilidades y roles, además de una ontología de obligaciones y compromisos por parte de los agentes involucrados. También en [6] los autores proponen un método, al que denominan ACA (*Accountability Centered Approach*), para la descomposición de responsabilidades en diferentes niveles y así establecer correspondencias entre las subresponsabilidades detectadas y las actividades de los procesos de negocio. Igualmente, también se ha explorado el uso de lógicas para describir aspectos tanto estáticos como de comportamiento dinámico de los sistemas. Un ejemplo es ALCQO(Q*) [7], que ofrece una extensión a la lógica descriptiva para incorporar el modelado de procesos y la representación de restricciones numéricas, a la vez que aprovecha la capacidad de razomiento de este tipo de marcos de trabajo basados en lógica. En cualquier caso, ninguna de estas propuestas desarrollan un marco conceptual completo cercano al modelado de responsabilidades, ni ofrecen la posibilidad de hacer uso de un DSL cercano al diseñador, con el que pueda representar el conjunto de participantes en el sistema y las interacciones entre ellos, así como analizar y validar el comportamiento del modelo desarrollado.

2.1 Objetos de responsabilidad

La falta de mecanismos para reconocer qué responsabilidad corresponde a qué agente pone de manifiesto la necesidad de disponer de una semántica más precisa para estos conceptos deónticos. Concretamente, para abordar esta carencia, una nueva especificación nos debe permitir fundamentalmente diferenciar claramente qué estados son posibles de aquellos que resultan de obligado cumplimiento. Además, estas situaciones posibles u obligadas han de poder ser asignadas, de manera explícita, tanto a uno como a varios agentes presentes en el sistema permitiendo, de esta manera, posteriores análisis de responsabilidad. Por tanto, se hace indispensable para tal fin que en la nueva especificación se mantenga siempre una referencia al responsable (o responsables) de cada acción de responsabilidad, como puede ser una obligación. También, el uso de obligaciones presenta otros aspectos a contemplar, como la temporalidad, según la cual las obligaciones pueden ser: *(a)* puntuales, si la obligación se descarga con una simple acción; *(b)* periódicas, si ésta requiere de una cierta acción cada cierto tiempo `t`; *(c)* temporales, similar a la anterior pero el instante `t` no es fijo, sino que queda sujeto a algún tipo de regla temporal más o menos compleja; o *(d)* continuas, que no se descargan nunca, sino que restringen en cada momento el comportamiento

del sistema. Obsérvese que una obligación continua en el tiempo puede ser vista como la prohibición de no hacer algo. Por ejemplo, las reglas: "*Estoy obligado a no fumar en todo recinto*" y "*Se me prohíbe fumar en todo el recinto*" modelan el mismo comportamiento. Este amplio abanico de posibilidades implica la necesidad de establecer mecanismos para controlar todo el ciclo de vida de las obligaciones, esto es, cuándo se crean, delegan o descargan.

Una de las ventajas más inmediatas al contemplar el tiempo en las obligaciones es que nos permite realizar de forma natural análisis sobre el grado de cumplimiento de dichas obligaciones en un instante temporal fijado. Estos análisis serán de gran utilidad para asegurar aspectos importantes como, por ejemplo, la calidad de los servicios proveídos por el sistema.

Una posible solución inicial para abordar esta problemática fue presentada originalmente en [8], donde los autores proponen un tratamiento orientado a objetos de las obligaciones. Así, permisos y obligaciones son materializados en dos tipos de objetos: *Permits* (Permisos) y *Burdens* (Cargas/Obligaciones). El modelado es como sigue: si un objeto tiene asociado un *Permit*, entonces tiene el permiso correspondiente para realizar una acción. En cambio, si tiene asociado un objeto *Burden*, tendrá la obligación de ejecutarlo. Por su parte, los objetos *Burden* se destruyen al descargarse de una obligación y también se pueden delegar hacia otro agente. Los permisos no se descargan, son extensibles, solamente se eliminan al ser revocados de forma explícita y también se pueden delegar.

2.2 Descripción del comportamiento del sistema

Desde un punto de vista formal, la materialización de los conceptos de responsabilidad se lleva a cabo mediante objetos del sistema que interacúan entre sí y con el resto de objetos. Además, su comportamiento se define en términos de *reglas* que indican las transiciones válidas entre estados.

Emplear lenguajes formales para representar el comportamiento y que sean capaces de gestionar la evolución de los objetos especificados bajo un modelo de tiempo, permitirá simular el comportamiento dinámico de un sistema como el descrito. Maude [4], basado en la lógica de reescritura, es un lenguaje formal ejecutable capaz de validar especificaciones orientadas a objetos y con declaraciones temporales. Es, por tanto, una herramienta muy eficaz tanto para la validación formal de la especificación como para la generación de prototipos tempranos ejecutables. Sin embargo, este tipo de lenguajes resultan complejos de aprender y gestionar por ingenieros no entrenados, lo que dificulta en gran medida su aceptación entre los arquitectos de negocio encargados de modelar las interacciones del sistema con otros agentes externos y usuarios. En este sentido, sería de gran ayuda en este campo disponer de DSLs gráficos que, por una parte, resulten cercanos al dominio de negocio tratado sin que, por otra parte, sacrifiquen los aspectos formales necesarios para validar la especificación o realizar su simulación. Posteriormente, a partir de estas especificaciones gráficas, la Ingeniería Dirigida por Modelos (*Model Driven Engineering*, MDE) ofrece mecanismos eficaces para llevar a cabo su transformación en artefactos de código, de forma transparente al arquitecto de negocio.

En concreto, en [3, 9] los autores hacen uso de la herramienta e-Motions[3], que permite definir el comportamiento de cualquier sistema visualmente y de forma intuitiva. En e-Motions, el comportamiento es expresado mediante un conjunto de reglas, cada una de las cuales identificará una transformación del estado actual del sistema, provocada generalmente por un acción. De forma concisa, cada una de las reglas se expresa de la forma: l :[NAC]* $\times$ LHS $\rightarrow$ RHS, donde l es la etiqueta de la regla; LHS (*Left-Hand Side*) y NAC (*Negative Application Conditions*) son patrones que expresan las precondiciones para que la regla sea ejecutada. Para que la regla l sea ejecutada ha de cumplirse el patrón LHS pero no deben satisfacerse ninguno de sus patrones NAC. e-Motions incorpora además duración a las reglas, permite el uso de expresiones OCL para calcular los valores de los atributos y resto de variables, y la posibilidad de representar como objetos de primer nivel a las acciones que se están ejecutando o se han ejecutado ya. Estos elementos (denominados *actions executions*) se usan por ejemplo para asociarlas explícitamente a las cláusulas NAC de una regla, lo que permite identificar acciones concretas que se encuentran en ejecución en un momento dado. Por otro lado, RHS (*Right-Hand Side*) es el patrón postcondición de la regla. La mecánica de la lógica de reescritura establece que, si se cumplen las precondiciones, la regla es disparada y el estado modelado en LHS es sustituido por el definido en el patrón RHS, trascurrido el tiempo definido como duración de la regla. Además, e-Motions cuenta con otros muchos conceptos para modelar diferentes tipos de sistemas, y con una semántica formal definida en términos de su representación en Maude [10].

3 Sistema de ventas online: caso de estudio

Como caso de estudio, esta sección muestra el modelado de negocio de un sistema de ventas online haciendo uso de la herramienta e-Motions. Para ello, un primer paso consiste en definir el metamodelo que declara los elementos que participan en el dominio, así como las relaciones entre los mismos (figura 1). Se identifican fundamentalmente tres tipos de elementos centrales en el sistema: Cliente (Customer), Servidor (Server) y Petición (Request). A su vez, los Servidores pueden ser clasificados en Vendedor (Vendor), Proveedor (Supplier) y Subcontrata (Subcontractor), de forma que cada uno de estos elementos tendrá una posición en el sistema, tal y como indican sus coordenadas xPos e yPos.

Desde el punto de vista del funcionamiento, en general, los Clientes realizan peticiones de productos a un Vendedor, que no puede satisfacer dicha petición por sí mismo, sino que ha de delegarla a uno de los Proveedores con los que trabaja. A su vez, el Proveedor podrá servir la petición o decidirá delegarla sobre una subcontrata. Cada uno de los agentes Servidores tendrá una capacidad máxima de peticiones (capacity) que es capaz de asumir simultáneamente.

El modelo inicial se construye a partir de una regla, denominada CreateInitialModel (fig. 6), cuyo patrón LHS aparece vacío, mientras que el patrón RHS

[3] `http://atenea.lcc.uma.es/e-Motions/`

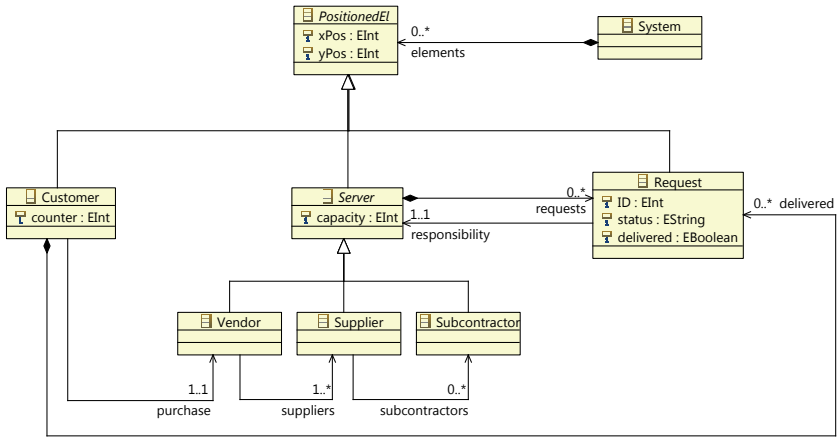


Fig. 1. Metamodelo del Sistema de Ventas por Internet.

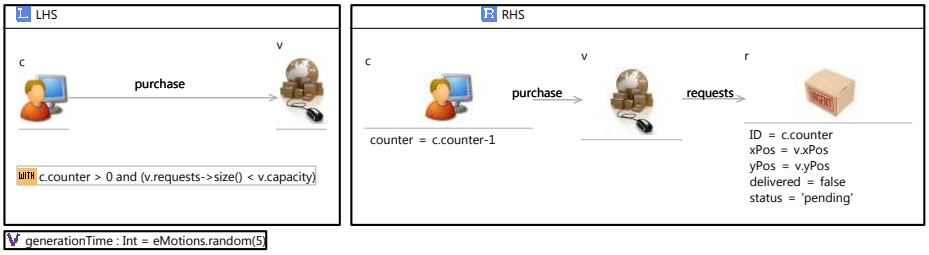


Fig. 2. Regla de generación de peticiones GenerateRequest.

modela el estado del sistema en el instante inicial. En concreto, para el caso expuesto, este estado inicial constará de un Cliente c que podrá realizar hasta un máximo de 25 Peticiones de productos al Vendedor v, ubicado en la posición (2,2). Dicho Vendedor trabajará con dos Proveedores, s1 y s2. Por otro lado, existen tres empresas subcontratadas, sb1, sb2 y sb3, de modo que s1 puede delegar en sb1 y sb2, mientras que s2 lo puede hacer en sb2 y sb3. Finalmente, ningún agente Servidor podrá gestionar más de 5 peticiones al mismo tiempo.

Una vez definidos los elementos del sistema y sus características, se modela el comportamiento del mismo en términos de reglas. La figura 2 muestra la regla atómica GenerateRequest, que modela el proceso de creación de peticiones de un Cliente a un Vendedor. Como se puede observar, existe una restricción OCL (*Object Constraint Language*) en la cláusula WITH de la regla, que restringe la creación de nuevas Peticiones por parte del Cliente si éste ya ha alcanzado su número máximo de Peticiones permitidas (c.counter) y, además, verifica que el Vendedor que recibe dicha Petición no ha alcanzado su capacidad máxima (v.capacity). Por otro lado, la variable generationTime consiste en un número

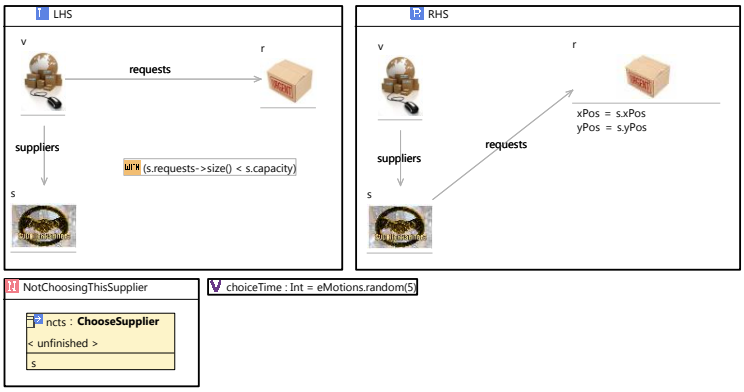


Fig. 3. Regla de selección de proveedor ChooseSupplier.

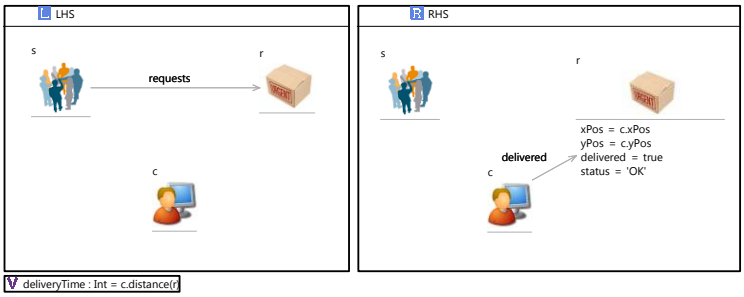


Fig. 4. Regla de envío de peticiones DeliveryRequest.

entero aleatorio definido en el rango $(0, 5)$, que establece el tiempo que consumirá esta regla durante su ejecución.

Son muchas las reglas que participan en el modelado de negocio del caso de estudio. Por ejemplo, obsérvese la definición de la regla ChooseSupplier, mostrada en la figura 3, que establece el modo en que se especifica la selección de un determinado Proveedor. En este caso el comportamiento es restringido por el patrón NAC, denominado NotChoosingThisSupplier, que define una acción (ncts) sobre el objeto s de tipo Proveedor. Dicha acción impide que, mientras la regla no haya finalizado, exista otra ejecución con el mismo Proveedor. De este modo se vigila que no se produzca una violación de la condición WITH de la parte izquierda, que controla que dicho Proveedor no acepte más Peticiones de las que puede gestionar simultáneamente (s.capacity). Al igual que antes, la duración de la regla se establece aleatoriamente dentro de un rango prefijado.

Finalmente, consideremos la regla DeliveryRequest, mostrada en la figura 4, que define el modo en que se hace entrega de una Petición realizada por un Proveedor. Nótese el uso de inyectividad en las reglas, como por ejemplo el objeto s de tipo genérico Server (que engloba tanto a Proveedores como para Subcontratas, véase la figura 1) evita tener que modelar otras dos reglas dife-

rentes, una para el envío de Peticiones realizada por Proveedores y otra idéntica pero para las Subcontratas. En esta ocasión, el tiempo de ejecución de la regla no es aleatorio, sino que vendrá determinado por el cálculo de la distancia entre la Petición r (con idéntica ubicación que el agente Servidor) y el Cliente final c.

4 Especificación de responsabilidades: caso de estudio

Una vez modelado el comportamiento básico del sistema, el siguiente paso consiste en incorporar distintos objetos de responsabilidad (véase sección 2). Nótese que en este caso de estudio se ha considerado además reducir el número de conceptos deónticos manejados, sin que ello implique una pérdida de expresividad. En este sentido, se mantiene que las prohibiciones no serán modeladas explícitamente, sino que serán reemplazadas por una definición equivalente: una prohibición es la obligación de no hacer algo.

4.1 Identificación de responsabilidades

En el caso de estudio presentado se detectan tres acciones susceptibles de ser controladas mediante objetos de responsabilidad: *(a)* enviar peticiones, *(b)* asumir peticiones y *(c)* delegar peticiones. Desde la visión deóntica, únicamente se hará uso de objetos de obligación y de permiso. Su comportamiento, específico para este dominio de aplicación, se detalla a continuación:

Permiso. Cada agente Servidor del sistema ha de tener permiso explícito asignado que le permita tanto poder hacerse cargo como enviar una Petición. Análogamente, para que un agente pueda delegar cualquier tipo de responsabilidad ha de disponer de los permisos pertinentes.

Obligación. Acción que establece una obligación de envío, antes de un determinado instante de tiempo, de una Petición concreta. Puede ser asignado a uno o a varios agentes Servidores. Se crearán de manera automática asignadas a cada nueva Petición resgistrada en el sistema.

4.2 Declaración de las acciones de responsabilidad

Al igual que ocurriera con los conceptos del dominio, los términos asociados a la gestión de responsabilidades deben ser igualmente metamodelados (figura 5). Así, los objetos de responsabilidad son especializados a partir de la metaclase AccountabilityAction (acción de responsabilidad), que posee dos atributos: tStart, que marca el instante de creación de la responsabilidad, y delegable, que indicará si dicha responsabilidad es delegable o no. Obsérvese que para poder delegar una responsabilidad se han de satisfacer dos condiciones: que el agente tenga asignado un objeto de delegación específico para dicha responsabilidad y que la responsabilidad en sí sea delegable. Una misma acción de responsabilidad puede ser asignada a uno o más servidores, que participan con el rol de agente (agent).

En concreto, las acciones de responsabilidad se especializan en Obligación (Burden) y Permiso (Permit). Las Obligaciones tienen un atributo, denominado

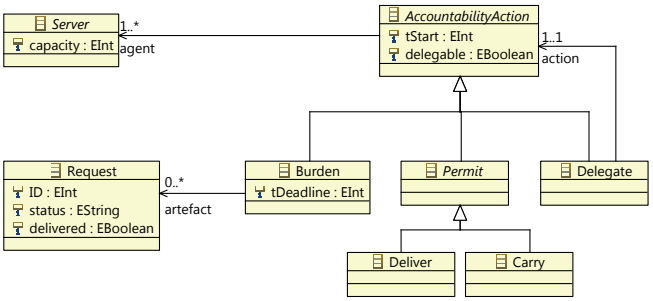


Fig. 5. Metamodelo de responsabilidades.

tDeadline, que determina el instante de tiempo máximo en el cual la petición debe ser satisfecha. En caso contrario, lo hará de forma tardía. Además, una Obligación está asociada a todas las Peticiones involucradas (artefact) en dicha responsabilidad. Por otro lado, la metaclase Permit se especializa a su vez en las acciones de Entrega (Deliver) y Transporte (Carry).

Nótese que en este dominio aparece una nueva acción asociada a la responsabilidad de cada agente: la Delegación, representada por la metaclase Delegate. En concreto, la Delegación consiste en el traspaso de algún objeto de responsabilidad (Permiso u Obligación) a otro agente del sistema. Este concepto, que no se ha extraído de la lógica deóntica, es sin embargo habitual en multitud de dominios de negocio y, por tanto, se ha considerado apropiado modelarlo explícitamente al mismo nivel que las acciones de responsabilidad anteriores. Finalmente, es importante resaltar que Delegate posee una asociación dirigida hacia la acción que se permite delegar (action). Así pues, se pueden dar tres tipos: delegación de *(a)* una Obligación, *(b)* un permiso de Entrega y *(c)* un permiso de Transporte.

4.3 Especificación del comportamiento mediante permisos

Una vez definidos los conceptos de responsabilidad de los que haremos uso en el caso de estudio presentado, el siguiente paso consiste en combinar dichos términos con los definidos en el metamodelo del dominio de negocio, descrito en la sección 3. La herramienta e-Motions permite la carga de varios metamodelos para definir un mismo DSL gráfico, lo que facilita la modularidad y reusabilidad de los metamodelos generados. Así, por ejemplo, el metamodelo de acciones de responsabilidad puede ser reutilizado fácilmente en cualquier otro dominio de negocio sin necesidad de alterar sus elementos.

Para la definición del comportamiento, primero hemos de modificar la regla que nos genera el estado inicial del sistema, incorporando ahora los objetos de responsabilidad en el patrón RHS (figura 6). En la tabla 1 se resumen los permisos asignados a cada uno de los agentes servidores del modelo.

Como se mencionaba en la sección 4.1, cada vez que se crea una nueva Petición en el sistema, automáticamente se genera una Obligación asociada a ella. Esto es modelado mediante una regla generateRequest, mostrada en la figura 7,

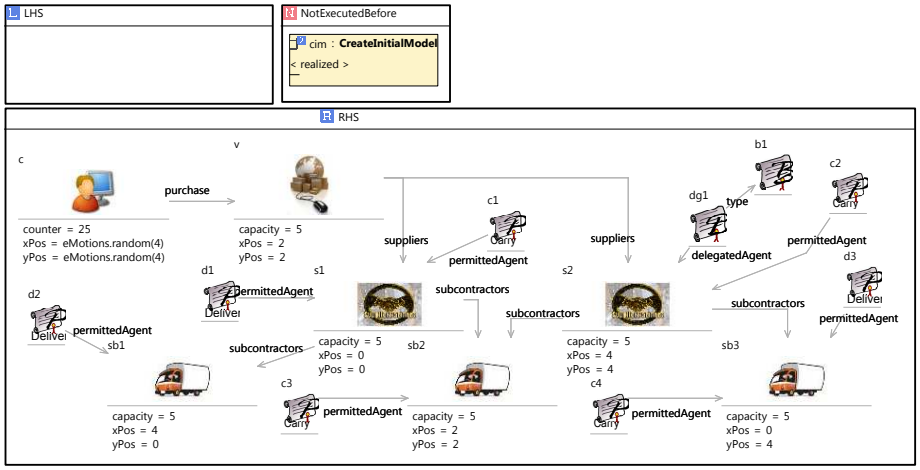


Fig. 6. Regla de creación del modelo inicial CreateInitialModel.

Objeto	Rol	Transportar	Entregar	D-Carga	D-Transportar	D-Entregar
s1	Proveedor	X	X	-	-	-
s2	Proveedor	X	-	X	-	-
sb1	Subcontrata	-	X	-	-	-
sb2	Subcontrata	X	-	-	-	-
sb3	Subcontrata	X	X	-	-	-

Table 1. Permisos asignados a objetos Servidores.

en la que se representa cómo ante la generación de una Petición r aparece una nueva Obligación b cuyo artefacto es r y cuyo agente responsable es el Vendedor v. Obsérvese que, mientras que agent puede variar durante el ciclo de vida de b (una Obligación puede pasar de un agente a otro), la relación artefact entre la Obligación y la Petición es invariante hasta la desaparición de b. Por consiguiente, este comportamiento del objeto de carga respecto a su agente responsable y su Petición asociada debe ser igualmente considerada en el resto de reglas. Además, el atributo tStart de la Obligación se inicializa al instante de tiempo en que se completa la ejecución de la regla y, por tanto, la responsabilidad creada comienza a considerarse activa. Análogamente, el tiempo en que expira dicha responsabilidad, tDeadline es inicializado a partir del valor de tStart, al que se añade un tiempo de vida desconocido. Finalmente, por motivos de simulación, la propiedad delegable se inicializa de forma aleatoria a un valor booleano.

Volvamos ahora a la regla ChooseSupplier, vista en la sección anterior (véase figura 3), a la que se incorporarán los objetos de responsabilidad, como muestra la figura 8. En esta regla, que modela la elección de un proveedor, se puede apreciar cómo el agente v, responsable de satisfacer la Petición (ver patrón LHS), delega su responsabilidad en el Proveedor s, tal y como se representa en el patrón

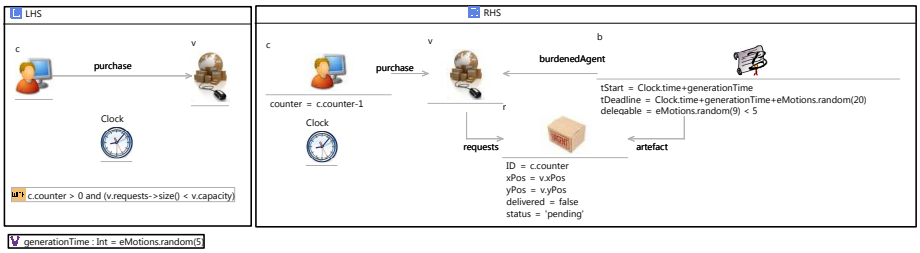


Fig. 7. Regla de generación de peticiones con permisos GenerateRequest.

RHS. Nótese que en esta ocasión aparece un nuevo objeto p de tipo Transporte (Carry), cuyo objetivo es imponer la restricción de que dicha regla sólo pueda ser disparada si el Proveedor s tiene el Permiso adecuado para asumir Peticiones. En caso contrario, esta regla nunca llegaría a ejecutarse para este Proveedor.

De manera análoga, redefinimos ahora la regla DeliveryRequest, originalmente representada en la figura 4 sin objetos de responsabilidad. Ahora, el comportamiento del objeto de Obligación b en el patrón LHS es idéntico al visto en la anterior regla de selección de Proveedor. Sin embargo, puesto que ésta modela el envío de la Petición al Cliente, en el patrón RHS dicha Petición desaparece del sistema, por lo que se considera concluído el ciclo de vida de un objeto de carga, iniciado en la anterior regla GenerateRequest. Para asegurar que sólo el agente adecuado s es quien participa en la regla, ésta sólo se se dispara cuando s dispone de los Permisos apropiados, en este caso, el Permiso de entrega Deliver. La regla se representa gráficamente en la figura 9.

Finalmente, obsérvese que el hecho de que en el modelo de negocio se permitan delegar ciertas acciones implica que deben aparecer nuevas reglas que regulen el correcto comportamiento del dominio. Un ejemplo es la regla DelegateRequest, representada en la figura 10. En concreto, esta regla modela la delegación de una

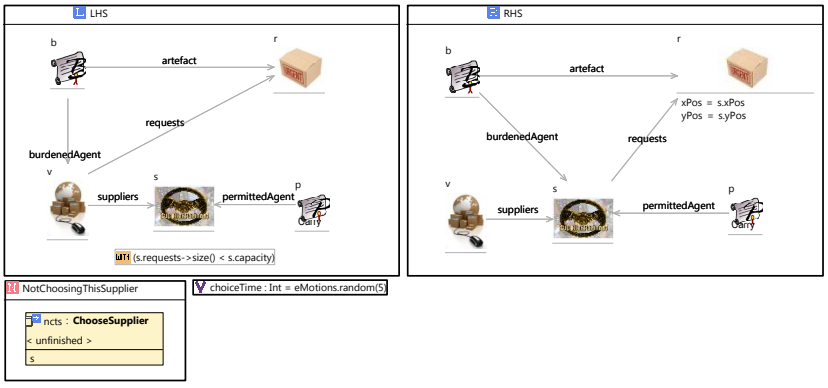


Fig. 8. Regla de selección de proveedor con permisos ChooseSupplier.

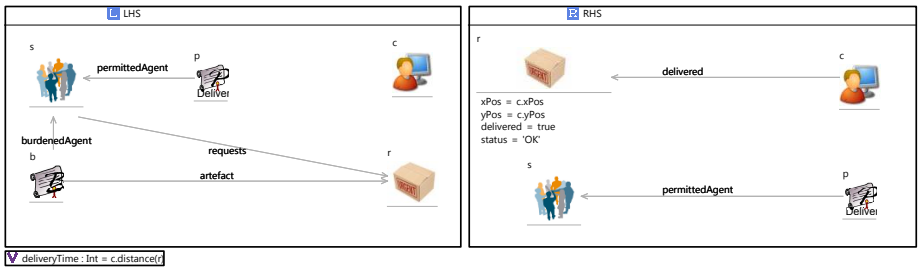


Fig. 9. Regla de envío de petición con permisos DeliveryRequest.

Obligación contraída por un Proveedor hacia un agente de tipo Subcontrata. En este caso resulta de interés analizar cómo se modelan los Permisos requeridos para que la delegación de una acción pueda hacerse efectiva. En el patrón LHS establece cómo el Proveedor s, que delega, requiere el Permiso adecuado para poder llevar a cabo la delegación de Obligaciones, esto es, tiene asignado un permiso de delegación (dg), que permite delegar acciones de tipo b1. Por otro lado, el agente sobre el que se delega (Subcontrata sb) debe disponer del Permiso necesario para cargar con las obligaciones que se le imponen. Para ello, la cláusula WITH de la regla fuerza a que la Obligación que se va a delegar sea efectivamente delegable y a que la Subcontrata que recibe la Petición no haya alcanzado su capacidad máxima (sb.capacity). Además, el patrón NAC, ejecutado sobre la el objeto actual r de tipo Petición, impide que se inicie una delegación de r si ya existe otra ejecución de la regla sobre dicho objeto.

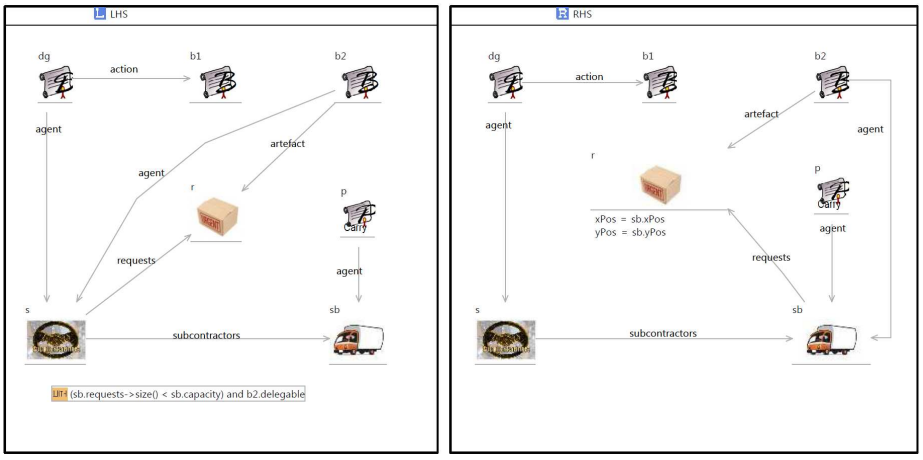


Fig. 10. Regla de delegación de petición con permisos DelegateRequest.

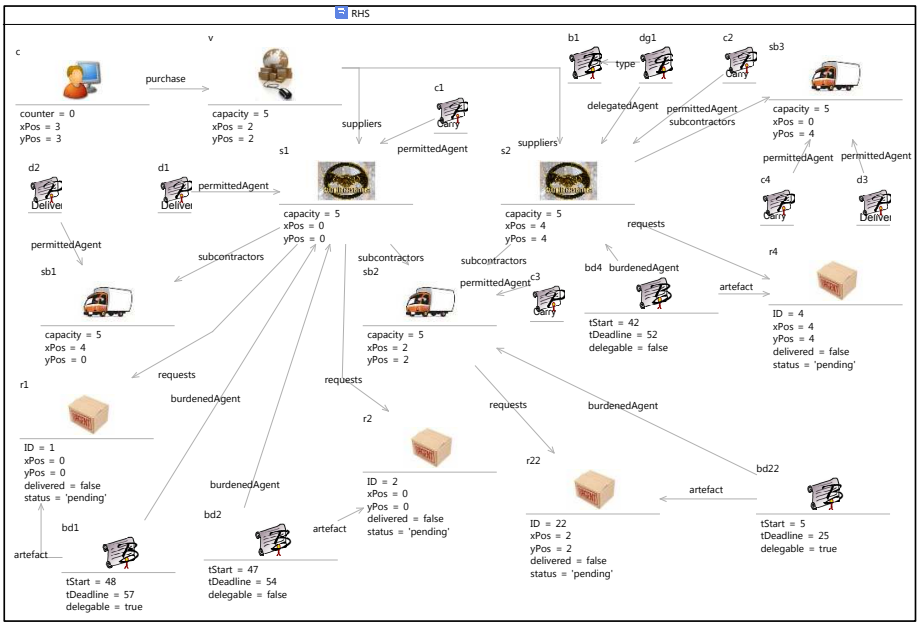


Fig. 11. Modelo resultante tras la simulación de 50 unidades de tiempo.

4.4 Simulación del Sistema

Una vez se dispone de las especificaciones del sistema, la herramienta e-Motions permite simularlo y analizar algunas de sus propiedades. En particular, es posible ejecutar el sistema partiendo de su configuración inicial (véase la regla CreateInitialModel en figura 6). La figura 11 muestra el estado del sistema de ejemplo después de 50 unidades de tiempo. Se puede apreciar que existen cuatro peticiones que no han sido satisfechas: r1,r2, r4 y r22, cada una de ellas con su objeto Burden correspondiente. En el caso de las peticiones r1 y r2, creadas en el instante 48 y 47 respectivamente, éstas aún no han sido enviadas. Por otro lado, r4 está adjudicado a s2 que no tiene permiso de envío, y únicamente podría delegar en alguna Subcontrata. Sin embargo, el Burden asociado a r4 impide que se pueda delegar y, por tanto, esta petición nunca será satisfecha, al igual que la petición r22 que pertenece a sb2 carente de permisos de envío.

5 Conclusiones y trabajo futuro

Disponer de lenguajes y herramientas para analizar y verificar los diseños de los sistemas basados en servicios se ha convertido en algo esencial debido a su creciente complejidad. Además, también es preciso contar con notaciones para representar ciertos aspectos de este tipo de sistemas hasta ahora no contemplados, como por ejemplo la responsabilidad de los agentes del sistema a la hora de

llevar a cabo sus acciones. Este trabajo presenta una propuesta inicial en la que hemos mostrado cómo es posible modelar dichas responsabilidades haciendo uso de la lógica deóntica, su reificación como objetos del sistema [8], y de lenguajes y herramientas de dominio específico.

Aunque el modelado de los permisos, obligaciones y delegaciones está completamente definido, hay varias líneas de trabajo futuro que nos gustaría acometer a continuación. En primer lugar es preciso validar la propuesta con casos reales que permitan evaluar su expresibilidad, usabilidad y escalabilidad. También es necesario definir el tipo de análisis que pueden realizarse con esta propuesta, así como su grado de aplicabilidad y utilidad para los usuarios finales (en este caso, diseñadores de alto nivel de grandes sistemas basados en servicios). Finalmente, estamos colaborando con ISO e ITU-T en la incorporación de estas ideas en alguno de sus estándares, y en particular en el que define el punto de vista de la Empresa de ODP [11], que precisamente se encarga de la especificación de grandes sistemas desde la perspectiva del negocio.

Agradecimientos. Los autores quieren agradecer a los revisores sus valiosos comentarios y esfuerzo. Este trabajo ha sido parcialmente financiado por los proyectos TIN2008-03107, TIN2008-00889-E y P07-TIC-03184.

References

1. ISO/IEC 10746, ITU-T Rec. X.901-X.904: RM-ODP. Reference Model for Open Distributed Processing, Geneva, Switzerland. (2010)
2. von Wrigh, G.H.: Deontic logic. Mind **60** (1951) 1–15
3. Rivera, J.E., Durán, F., Vallecillo, A.: A graphical approach for modeling time-dependent behavior of DSLs. In: Proc. of VL/HCC'09. (2009) 51–55
4. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework. Volume 4350 of LNCS. Springer (2007)
5. Zou, J., Pavlovski, C.: Towards accountable enterprise mashup services. In: Proc. of the IEEE International Conference on e-Business Engineering, Hong Kong, IEEE Computer Society (2007) 205–212
6. Tseng, M., Chuan, J., Ma, Q.: Accountability centered approach to business process reengineering. In: Proc. HICSS'98, Hawaii (1998) 345–354
7. Liu, H., Qing, L., Gu, N., Liu, A.: Modeling and reasoning about semantic web services contract using description logic. In: Proc. of the 9th International Conference on Web-Age Information Management, China (2008) 179–186
8. Linington, P.F., Neal, S.: Using policies in the checking of business to business contracts. In: Proc. of POLICY 2003. (2003) 207–218
9. Troya, J., Rivera, J.E., Vallecillo, A.: Simulating domain specific visual models by observation. In: Proceedings of symposium on theory of modeling and simulation. DEVS 2010. Orlando, Florida, USA. (2010) 46–53
10. Rivera, J.E., Durán, F., Vallecillo, A.: On the behavioral semantics of real-time domain specific visual languages. In: Proc. of WRLA'10. Number 6381 in LNCS, Springer (2010) 174–190
11. ISO/IEC 15414, ITU-T Rec. X.911: Information Technology — Open Distributed Processing — Enterprise Language. (2006)

Sesión 6: Fundamentos de servicios

SEMTOUR-Studio: A Semantic Web Services Creation Tool for the Tourism Sector.

F.J. Lacueva (fjlacueva@ita.es), M.A. Barcelona (mabarcelona@ita.es), L.Garcia (lgarcia@ita.es), E.Mendoza (emendoza@ita.es), J. Lalana (jlalana@ita.es), J. Vea-Murguia (jvea@ita.es)

Instituto Tecnológico de Aragón, María de Luna 7-8, 50018, Zaragoza, Spain

Abstract. SEMTOUR-Studio is a semantic Web Services composition editor developed to allow the population of the SEMTOUR Tourism Web Services Platform (STWSP). On the one side SEMTOUR Studio allows to deploy basic Web Services within the STWSP: already existing services in external systems can be "normalized" (semantically annotated, grounded and BPEL wrapped), in order to make them deployable and consumable within the STWSP. On the other hand it allows creating new virtual services by the composition of already deployed services.

Keywords: SWS, SAWSDL, BPEL, WSMO-Lite, OWL, XSLT, Grounding, SEMTOUR, Service Composition, e-Tourism, Virtual Enterprises.

1 Introduction.

The SEMTOUR project [1] took the results of the Composetour [2] project to create a Tourism Value-Added Web Services Platform. The platform has three main components:

1. The Community of Service Users: travelers who want to make a trip, who are registered with in the community and who can give their opinion about the consumed product and the Web Services used to contract the product.
2. The Semantic Web Service Platform, where services are deployed, executed, monitored and evaluated, both by the user and by the Platform (liability, response time, …)
3. The SEMTOUR-Studio, which is introduced on this paper, a semantic Web Services composition editor developed to allow the population of the SEMTOUR Tourism Web Services Platform (STWSP).

Fig 1 introduces the system architecture of the SEMTOUR Platform. The main goals of the SEMTOUR project are:

- Populating the infrastructure with touristic semantic Web Services: already existing services which are normalized, that is semantically annotated, grounded and BPEL wrapped, in order to be consumed within the SEMTOUR Platform.

- Creating and establishing the community of users for the SEMTOUR Platform which evaluates and grants reputation about the consumed products and services.
- Creating an algorithm for service discovery based on the reputation of the services (or of the products they provide) and on its QoS.
- Making the creation easier of "virtual" travel agencies by the composition of "SEMTOUR Services".

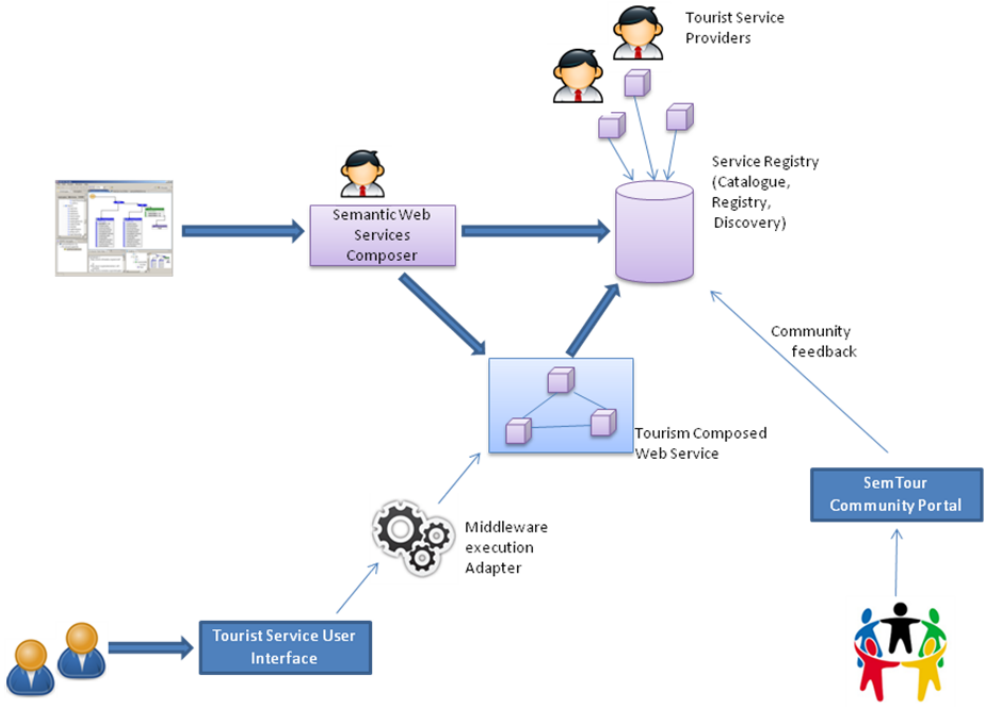


Fig. 1. Conceptual System Architecture of SEMTOUR.

As we previously mentioned, this paper introduces SEMTOUR Studio, a semantic Web Services composition editor developed to allow the population of the STWSP. On the one side SEMTOUR Studio allows to deploy basic Web Services within the STWSP: already existing services in external systems can be semantically annotated, grounded and BPEL wrapped, in order to make them deployable and consumable within the STWSP in a standard way, such as what we call SEMTOUR Services.
On the other hand, the SEMTOUR Studio allows the creation of new virtual services by the composition of already deployed services. It allows the selection of the existing services both by their semantic description and by their reputation and the reputation of the products they provide.

The rest of the paper is dedicated to summarize our work within the project. It starts by the definition of the requirements of the ECSWS, it follows with the

definition of its architecture and the interfaces with the SEMTOUR Platform and it concludes with the implementation and test of the solution.

2 SEMTOUR Studio Requirements.

We briefly introduce here the Requirements of the ECSWS, which are gathered in the deliverable E3.2 [3] of the SEMTOUR project. This deliverable was approved by the project consortium after their revision and comments inclusion.

As we already mentioned in the introduction paragraphs, main objectives of the ECSWS are to allow the population of the SEMTOUR Tourist Web Services Platform (SWSP) and the creation of new services by composing SEMTOUR Services. Both kinds of services, either Basic Services or Composed Services, have to be described by (annotated against) a WSMO-Lite [4] ontology and their grounding processes have to reference a community shared ontology and - virtualized by a BPEL [5] process.

Additional high level requirements are: integration with the platform security in order to avoid that anyone could publish, consume or create SEMTOUR Services; and the possibility of filter discovered services by both functional and not functional (reputation, QoS) criteria.

From these high level requirements we refined the functional and non functional as are summarized in next paragraphs.

2.1 Functional Requirements.

Next table summarizes the functional requirements of our tool which are more important for the aim of this paper.

FR id.	Requirement
FR0003	High Level Functionalities of the ECSWS would be: a WS annotation tool for annotating WS, a tool to access to the SWS repository; a tool to access to the Ontology Repository; an ontology visualization; a service composition.
FR0005	"Files" to define the different elements to use should be based on XML derived languages such as OWL, WSMO-Lite, BPEL or XLST
FR0012	The user should be able to obtain the ontology definition, the security and reputation policies and the services deployed in the community.
FR0018	The local WSDL files that syntactically describe the WS should be annotated against the SEMTOUR community ontology.
FR0022	Each of the WS will produce a Basic Business Process
FR0023	Business Processes also can be created as the composition of services already deployed in the community.
FR0025	The data to use within the composed BPs should be concordant with the Domain ontology.
FR0027	The code to be generated should include all the information necessary to fulfill the security policies, execution log, …
FR0030	The composed BPs should also be annotated against the community ontology. As far as possible it should be performed automatically.

FR0031	In order to allow the consumption of the SEMTOUR community services from outside the community, their WSDL files should be made public available.

Table 1. Relevant Functional Requirements.

2.2 Non Functional Requirements.

Next table summarizes the Non Functional Requirements of the ECSWS.

NFR id.	Requirement
NR0001	WSMO-Lite will be used to annotate SWS
NR0002	The SEMTOUR Studio should be able to be used in different hardware and software platform.

Table 2. Relevant non Functional Requirements.

3 Another Editors Initiatives.

Semantic Web Services, SOA, (Dynamic) Process Compositions and BPM are going to be the research items which will be the foundations of the future enterprises. As a consequence; there have been a lot of initiatives which worked in trying to put all together to work in the achievement of the Future Internet realization.
Before start working in the SEMTOUR Studio we perform a state of the art of similar tools which already exists or which are under development. WSMO Studio [6], INFRAWEBS [11, 12], SOA4ALL [9] and SUPER [10] are examples of the studied tools.
Some of their libraries and tools are being used and adapted for implementing our ECWSW. The details about the modifications or additions included by us will be explained in the section 5 of this paper.

4 SEMTOUR-Studio Architecture.

Once we defined the requirements of the ECSWS we proceed to design it by defining its architecture. On the one hand it has to fit the defined requirements; on the other hand it has to correctly assemble with the other components of the SEMTOUR Platform architecture [14]. To get it we proceed to define a top-down module definition as it figures on deliverable PT3.E2 of the SEMTOUR project [13]. We briefly introduce the architecture and the use cases to be implemented on the next paragraphs.

4.1 Use Case to Be Implemented.

In [15] Papazoglou classified Web Services in Basic Services and Composite Services, in consequence we decided to implement two use cases: the Basic Service

annotation, grounding and "BPELLING" processes and the Definition of Composed Services (by SEMTOUR deployed services aggregation).

We introduce both use cases in next paragraphs and they are described on Figure 2 included under the "Create BP" use case.

Deployment of Basic Services.

In order to deploy an existing Web Service within the SEMTOUR community it has to be annotated, grounded and coded ("BPELLED"). These activities are included in the "Create Basic BP" of the previous diagram.

In [16] they define the way in which WSDL files [17] describing Web Services should be annotated using the SAWSDL [18] extension. From [16] it can follow that the semantic description of an existing Web Service consists of two steps (which have not necessarily to be performed in this order):

- The WSDL file is annotated against the ontology (the domain + WSMO-Lite ontology) in order to classify the operations, kinds of messages and data types. The resulting file is an SAWSDL file.
- The XML schemas of the original WSDL file are used to create the Grounding transformation files [22] with respect to the community shared ontology: Lowering when data are used to invoke a service operation and Lifting for the returned data.

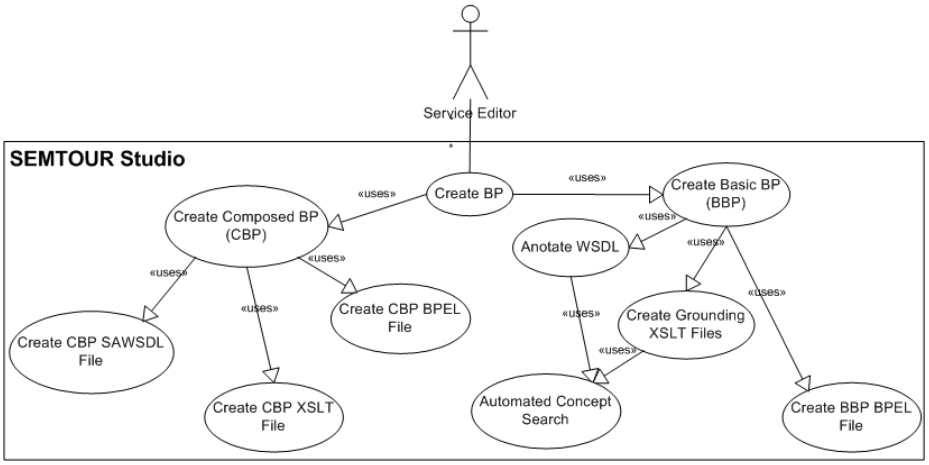


Fig. 2. SEMTOUR Studio use cases.

As the resulting XSLT files of this processes has to be referenced by the resulting SAWSDL file of the previous step, it has no sense to determine in which order it has to be performed. But, if some automatic annotation would be possible, as we explain after, perhaps we can consider that the presented order may be "more" suitable.

The final step is the creation of the code to be used within the community in order to consume an external existing WS. In our case, we use BPEL as the Business Process language [23]. The BPEL capability of subscribing to different kind of messages is used in order to give access to the different operations a WSDL file may content. The generation process of the BPEL code automatically includes the invocation to the original Web Service operation and the invocation to the transformation processes based on the XSLT files generated on the Grounding step.

Proceeding on this way, every Basic Service deployed in the community, that is a SEMTOUR Basic Service, will be always accessed in a standardized way. To get it some automatic changes have to be performed in the SAWSDL file in order to operate in a correct way:

- The XML schemas of the type definition have to be changed to the ones representing the ontology. The grounding processes transform from the ontology data representation to the original service data representation, the XML schemas defined in the original service WSDL file (Lowering), and vice versa (Lifting). In consequence, the input and output data of the exposed operations to the SEMTOUR community fulfill the XML schema representing the ontology.
- The endpoints of the original services have to be changed to the ones through the SEMTOUR Platform is going to offer the SEMTOUR service wrapping the original service. When we are creating a Basic Service, one of the processes is the creation of a BPEL file. This file has to be executed within in the community by a BPEL engine which will make the operations accessible in a proprietary endpoint. This is the endpoint which should appear within the SAWSDL file describing the service to the community.

Although the first change can be performed either by the editor or by the community platform, the endpoint change can only be done once the BPEL is deployed in the execution engine that is when the SEMTOUR service endpoint is known. In consequence we consider that it is better to be implemented by the SEMTOUR Platform.

Composed Services Development/Deploying.

Composed SEMTOUR services are created by the aggregation/orchestration of services which have already been deployed in the SEMTOUR community. SEMTOUR services are, at least from the input and output data point of view, standardized processes: every service deployed within the SEMTOUR Platform should have inputs and outputs which are instances of the SEMTOUR ontology. That will help in some aspects of the creation of the SEMTOUR composed services.

In the first place, as all the data of the input and output messages of the operations of a SEMTOUR service must be instances of the domain ontology, when we want to invoke an operation of a SEMTOUR service, it will not need to make any transformation, in consequence grounding transformations would not be needed. Even that, and in order to make everything as standardized as possible, when we create Composed SEMTOUR Services we decided to invoke the identity Lowering and Lifting functions.

In the second place, the automation can be applied to the BPEL generation too. If the SAWSDL file of a given SEMTOUR service would contain the choreography of the original service, for example using the WS-DCL [21] as suggested [22], it will not be very difficult to create the orchestration of their operations. Even that, our first approach to the implementation of the composed services is to detect the matching between the output and the inputs of the operations of a given SEMTOUR service. This matching process also allows us to detect the order of the services under composition.

With this approach our intention is to start working in the integration of the SEMTOUR Studio with the SEMTOUR Platform. We are aware of the limitations of this approach but it will allow us to have a look into the problems we need to solve in order to advance on the road to the dynamic service composition. Moreover, even if the choreography of the services is already established, the editor should consider the user interaction in order to give him the opportunity to create its own service, for example, to include the invocation to log services.

In order to make the composed services consumable they have to be published by creating its SAWSDL file and deploying it together with the BPEL file and the grounding files, in the same way as a Basic Service is published. Again, the creation of the SAWSDL file can be done automatically. The composed service is going to have just one operation, resulting from the orchestration of the operations of the used/included SEMTOUR services. For this operation, the input and output messages can be obtained from the input of the first invoked operation and the output of the last one.

The user interaction will be only needed to determine the name of the service and to annotate the elements of the WSDL describing the composed service. If we try to name or to annotate the new service/operation automatically we can consider two ways to do it. The first one is to annotate the elements of the composed service WSDL with the annotations of the correspondent WSDL elements of all the source services. The second one is by trying to make it automatically by searching the matching between the name of the SAWSDL elements and the ontology concepts just as we do with during the Basic Service annotation and grounding definition process. Even that, from our point of view, as it is not possible to build a complete ontology dictionary, the interaction with the user is necessary, although it can be reduced to the minimum.

Finally, as it happens with the Basic Service, the endpoint should be changed before the new service could be consumed within the community, that is, during the deployment process.

4.2 ECSWS High Level Architecture.

SEMTOUR Studio is the interface of the SEMTOUR Platform to the actual services owners. As we already mentioned in previous paragraphs its objective is to allow non technical staff to share their business services (Basic Services) with the SEMTOUR community and to create new business process from the composition of already

deployed SEMTOUR services composing new business processes using already deployed services.
Figure 3 shows the architectural definition of the SEMTOUR Studio Editor and the relation of each of its component with the rest of the components of the SEMTOUR Platform. We briefly introduce these components:

- The concerns of the ST_ECSWS_SemanticManager are the manual and semi-automatic semantic annotation of the Web Service description (WSDL) against the Tourism domain ontology and the WSMO-Lite ontology. As a consequence it has to interact with the community to obtain the OWL ontology definition and the set of ontology concepts in order to get the service semi-automatically annotate.
- Once the services has been semantically annotated, the next step is to standardize the data they exchange with the rest of the community, that is to define their Lowering and Lifting (XSLT) mapping processes. The ST_ECSWS_GroundingEditor is created to do it. Although some automatic process can be performed in the case of the Grounding it has to be mainly performed manually.
- From the development team point of view every "executable" element should have to be deployed in the SEMTOUR Platform in a standard way. All the services, either basic or composed, have three elements: one SAWSDL file containing their semantic description in terms of the community ontology; one grounding file containing the Lifting and Lowering XSLT schemas to transform their data; and one BPEL file containing the code to access to the actual services. Obviously the ST_ECSWS_SemanticBPEditor is in charge to create automatically the BPEL wrapper of the Basic Services, and to allow users to create composed services by glueing already deployed services.
- Finally the ST_ECSWS_LocalRepositoryManager is in charge to hide the exchange of information between the computer in which the ECSWS is being executed and the community repositories which are represented by the ST_ECSWS_RemoteRepositoryManager in which services are deployed.

Although most of the interactions of the editor's components are designed to be done through the repository manager's components, some of them can be done directly against some of the SEMTOUR platform modules. For example, the ST_ECSWS_SemanticBPEditor allows the validation of the code of a composed service under development against the ST_ECWSWS_ExecutionEngine in order to avoid as many errors as possible, before the service deployment.

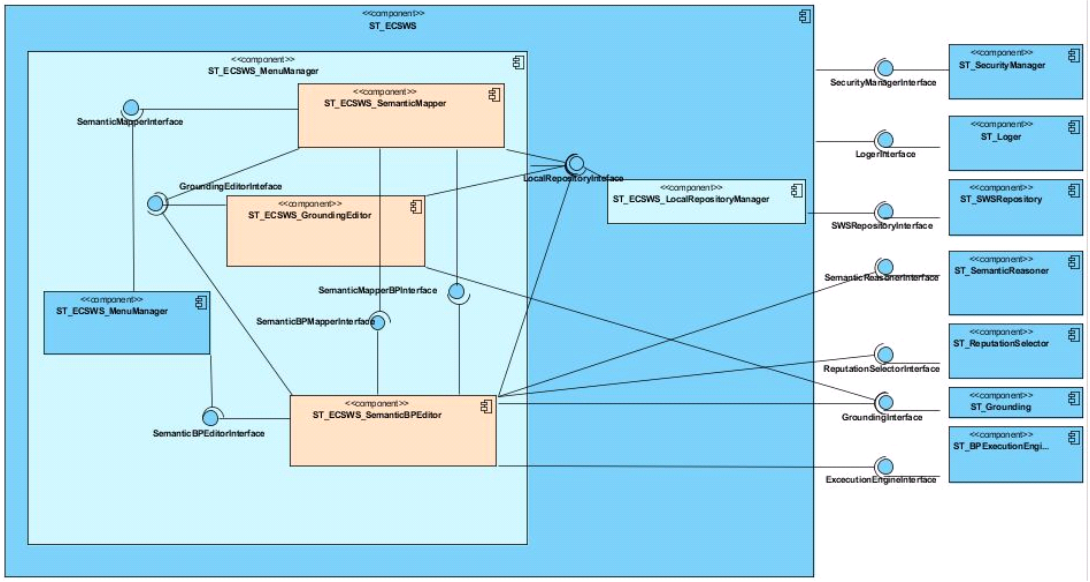


Fig. 3. SEMTOUR Studio Architecture.

5 ECSWS Development State.

By the time this paper is being written we are still developing the ECSWS. It is going to be released by July of 2011 in order to be tested and integrated with the rest of the SEMTOUR Platform. From the previous components diagram, we are currently working in the Grounding Editor and in the Semantic Business Process Editor. Next paragraphs summarize the adaptation of the Open Source Modules we are integrating and some limitations related to the implementation we assumed because of the scope of the project.

At the moment, apart from the integration test to be performed with the two other components presented in these paragraphs, ST_ECSWS_SemanticManager component is already finished and integrated within the ST_ECSWS_MenuManager. The other two components are still under development. The ST_ECSWS_GroundingEditor is close to be finished while the ST_ECSWS_SemanticBPEditor has already passed the Ecuador of its implementation: the Basic SEMTOUR Services is already released while the Composed Services are still under development.

5.1 ST_ECSWS_SemanticManager Development State.

The ST_ECSWS_SemanticManager has been already released. It has adopted some components from WSMO Studio modeling environment, modifying them in order to correctly generate SAWSDL files annotated against WSMO-Lite ontology instead of against WSMO ontology. We have added an automatic annotation process

based on the capability to associate synonyms to ontology concepts which was introduced by Bonino et col. in [23, 24, 25, 26].

We adapt some of the DOSE [25] libraries in order to allow the automatic annotation of a WSDL file. We developed an OSGI bundle, just another kind of software services prepared to be deployed on an OSGI runtime framework. Between other, the bundle has an operation which receives two input parameters: the first parameter is the URI of a file containing the concepts of the ontology together with their synonyms (which can be expressed in different languages); the second parameter is the word for which the meaning within the SEMTOUR community is search. If the meaning of the word is found (it exists a concept within the ontology having the searched word in its synonym list) then the URI of the concept representing the meaning of the word is returned in order to be used to annotate the WSDL file.

We use this bundle for trying to reduce the manual annotation of the WSDL to the minimum. Our method proceed on this way: we take the names of the operations, services and datatypes elements of the original WSDL file and we iterative invoke the bundle operation parameterized with the element name together with the file of synonyms of the ontology. For each of the names of the WSDL elements, if an URI is returned, we include it within the element annotation of the SAWSDL file under creation.

Although this process can help to reduce the annotations to be written by hand, the quality of the results is close related to the synonyms file content and, obviously the human interaction will be needed to revise the annotations and to add some references which cannot be automatically done. For example, consider the different ways to write the concept “room booking”. It is easy to imagine this concept write like: “roombooking”, “room_booking”, “reserva de habitación” or “reservahabitacion”… within WSDL files describing services or operations for booking a room. What is not obvious but is possible is that “RB”, “xy” or “reserva” could have the meaning of “room booking” within a WSDL file.

We want to mention that in the development of the Semantic components we have used the easyWSDL and easySAWSDL libraries in order to create and validate the different WSDL files [27]. Both, together with other useful libraries, have been developed under the SOA4ALL project [9].

5.2 ST_ECSWS_GroundingEditor

As we have already mentioned, we try to automatically annotate the data types. It is just a part of the work it has to be performed in order to make WSDL data and schemas semantically available for the SEMTOUR community. The second part is the creation of the Grounding files and their references in the SAWSDL file. As we introduced in paragraph 4.2, it is going to be implemented by the ST_ECSWS_GroundingEditor. Once the SAWSDL has been annotated we proceed to create the Lowering and Lifting files. The matching between the XML schemas has to be performed by hand. The results of the matching processes are the XSLT files which are going to be used to transform the XML inputs and outputs of the

semantic SEMTOUR Services or their operations to the inputs and outputs data formats of the original services.

In order to make the implementation of the matching process easier, we automatically create a XML Schema Definition (XSD) file containing a data structure definition equivalent to the instance structure of the OWL ontology. Proceeding on this way, the creation of the grounding files is reduced to the problem of creating XSLT files which transforms XML fulfilling an XSD, to XML data fulfilling another XSD. Obviously, we create different files to transform the input (Lowering) and output (Lifting) data. We use the OWL2XSD classes of the OWL2WSDL[29] library in order to automatically create the XSD file which is equivalent to the ontology instances.

We know that the transformation of the OWL entities structure to an XML Schema could suppose some semantic lack. For example, OWL allows defining the Accommodation class as the union of the Hotel, BB, Refuge, GuestHouse … classes, which is not possible using an XML Schema. From our point of view, this lack of semantic is not important for the problem we are solving. On the one side, the SEMTOUR Services are Web services and they can be invoked both from the SEMTOUR community and outside by external customers. So the input and the output parameters of a SEMTOUR service should be standard, they are XML data, and as SEMTOUR service, fulfills the XSD of the ontology. On the other side, the invoked Web services receive and return XML data fulfilling the XML Schemas declared within its WSDL file. Briefly, they are using XML data.

If within a SEMTOUR Service there is the necessity of transform XML to ontology entities, for example to choose a hotel according to with the requirements of customer, the data should have to be transformed to ontology instances, this instances should have to be added to the ontology and the ontology with the instances should be sent to a reasoner together with the corresponding description logic query. In other words, an additional service should have to be implemented by the SEMTOUR platform and additional code should have to be automatically generated during the Basic Process generation, to invoke this new infrastructure service. This additional code can be used, for example, for ordering the returned values following additional criterias such as the quality and reputation of the returned products or the provider service.

5.3 ST_ECSWS_SemanticBPEditor Developent State.

The ST_ECSWS_SemanticBPEditor component aims to allow a user to create the BPEL code which implements either a Basic SEMTOUR service or a new Composed Service.

The Editor of Basic SEMTOUR Services.

After annotating the WSDL, generating the Grounding files and include reference to their URIs in the WSDL file, the BPEL for a Basic SEMTOUR service can be created.

Once the the SAWSDL file has been created, it is possible to create a "BPELED" wrapper which will allow to access the original Web Service through it: it will contain a message subscription for each of the new messages, their data will be compliant with the XSD of the ontology; the message subscription will invoke the Lowering transformation of the input data; the returned data will be used to invoke the original Web Service; finally the returned data will be transformed to the ontology format by invoking the corresponding Lifting transformation.

Finally, as we describe in paragraph 4.1, before the new Basic Service can be deployed two additional changes should have to be performed in its SAWSDL. The first step can be either realized within the ECSWS or by the remote repository manager. This step consists in the substitution of the original WSDL data schema definition by the corresponding to the ontology and extending this change to all the messages definition. The second change can only be performed within the community as it concerns with the replacement of the original endpoint by the one that corresponds to the server in which the BPEL is going to be deployed. Obviously this change can only be performed once the service is deployed.

The Editor of Composed SEMTOUR Services.

Composed services are created by the aggregation of the operations of already deployed SEMTOUR Services (either Basic or Composed). To create them we proceed in opposite direction to the one we follow with Basic Services. We first generated the BPEL, and then we generated the SAWSDL file which will be used to publish and to discover the new service once it would be deployed. As the services to be composed are SEMTOUR Services, they all will receive standard messages and in consequence the Grounding files should not have to be created (if needed they can be always created as the identity function).

The creation of the new service starts with the selection, by the user, of the services and their operations which are going to be aggregated and orchestrated by the new BPEL coded. Once they are selected, an automatic process is run to match inputs and outputs of the services.

6 Conclusions and Future Work.

The authors of this paper are members of the eLogistica work group of the Instituto Tecnológico de Aragón. Our interest is to research in the way ICT (both hardware and software) can be used to model, simulate and improve enterprise processes. Supply Chain Logistic Processes is our main sector of application as its processes are a well known field of exemplary synergies between the real world processes and the virtualized one. Although Supply Chain logistic is an historic field of use of the ICT, the popularization of use of the ICT facilities had created the actual trends to apply similar ideas within the eTourism, the eHealth or the eLearning sectors between many others.

The extension and generalization of the use of the ICT to the proposed or to others, to improve their business processes, can easily be reached. It depends on the existence of software systems supporting the different stages of an application creation (definition,, implementation, management and maintenance). This systems should be abstract enough to separate the sector (the domain data and processes definition and description) from the basic infrastructural services needed to create the business ecosystems (service publishing, discovering, execution, monitoring, …). And they can take advantage of the use standards interfaces to describe the services (WSDL), to allow the interoperability between services, between the services and the infrastructure (XML) and to aggregate them in new services (BPEL)

The SEMTOUR platform is an example. If we replace the domain ontology definition, the tourism one, by and eHealth ontology and we populate it with Basic eHealth Services, we will have the SEMHEALTH platform and the SEMHEALTH Studio. Obviously some improvements have to be made in SEMTOUR Studio and in SEMTOUR platform before their use can be generalized.

Some improvements to SEMTOUR Studio have been mentioned on this paper. An example is the inclusion of code for transforming XML data into OWL entities in the BPEL, useful if some reasoning is necessary with the returned data of an invoked service or service operation. Another example is the inclusion of facilities to define the security policies which the service follows and, in the case of Composed Services, the service demands to their providers.

The former is one of the possible improvements to make in order to implement Dynamic Service Composition both in the BPEL generation and in its execution. Another possible improvement is the heterogeneity of business processes to perform and action. For example, consider the creation of a composed service for making the reservation to travel from Madrid to Geneva spending three nights in Paris and resting a week around Geneva before coming back to Madrid. Suppose there are two candidate hotel in Paris A and B. The reservation process of A proceeds as follow: the process of answering the user data, the date of arrival and the date of departure, check the availability, answer for the confirmation and offers the payment process. The reservation process of the Hotel B receives the user data, the check in date and checkout date, request the confirmation and start the payment process. With this scenario, how can a virtualized process to create the full trip be created?

The answer seems to be clear. As with the ontology it is a two steps process: first, the knowledge about the domain processes should have to be expressed in pattern processes representing the operations work flows; next, the WSDL service description should be improved to include the matching of their operations to the standard processes and/or processes operations. In some cases, this will suppose the choreography definition to orchestrate two or more service operations in order to perform one service pattern operation.

For us, SEMTOUR Studio, is the second step of a bottom up approach to build a P2P semantic service integration platform. We started by defining the reference architecture and making an implementation of openBIP [32]. Next steps are to improve the discovering capabilities of openBIP with semantic search and to include BPEL engine. It would allow us using SEMTOUR Studio as the user tool to deploy

services within openBIP. They are steps on our road to make software services facilities of the virtual world.

7 References.

[1] Semtour Home Page. http://www.ines.org.es/Semtour/
[2] Composetour Home Page. http://www.ines.org.es/composetour/.
[3] F.J. Lacueva, P.Peña, J.Vea-Murguia, E. Mendoza, S.Bilbao. Semtour. E3.2 ECSWS Requeriments Document.
[4] WSMO-Lite. http://www.wsmo.org/ns/wsmo-lite/
[5] Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf
[6] Marin Dimitrov , Alex Simov , Vassil Momtchev , Damyan Ognyanov. WSMO Studio - an Integrated Service Environment for WSMO (2005). In Proceedings of the 2nd Workshop on WSMO Implementations (WIW 2005). Volume 134.
[7] WSMO WG. Home page. http://www.wsmo.org/index.html
[8] Conceptual Model for Services. Home page. http://cms-wg.sti2.org/
[9] SOA4ALL. Home Page. http://www.soa4all.eu/
[10] SUPER. Home Page. http://www.ip-super.org/
[11] INFRAWEBS. http://www.infrawebs.eu/
[12] Agre G. INFRAWEBS designer - A graphical tool for designing semantic Web Services [Internet]; 2006 [cited 2011 Apr 18]. Available from: www.scopus.com
[13] F.J. Lacueva, P.Peña, J.Vea-Murguia, E. Mendoza. PT3.E2. ECSWS Design Document.
[14] S. Bilbao, PT3.E2. SEMTOUR Platform Architecture Design Document.
[15] M.P. Papazoglou, D. Georgakopoulos, Service-Oriented Computing, special issue, guest editors introduction, Commun. ACM 46 (10) (2003) 24–28.
[16] T. Vitvar, J. Kopeck y, J. Viskova, D. Fensel. WSMO-Lite Annotations for Web Services. In the Semantic Web: Research and Applications, ESWC 2008.
[17] Semantic Annotations for WSDL Working Group. http://www.w3.org/2002/ws/sawsdl/
[18] Web Service Definition Language (WSDL). http://www.w3.org/TR/wsdl
[19] M. Dimitrov, A. Simov, M. Konstantinov and V. Momtchev: WSMO Studio - a Semantic Web Services Modelling Environment for WSMO (System Description). In ESWC 2007, Proceedings of the 4th European Semantic Web Conference, June 2007, Innsbruck, Austria
[20] D. Martin, M. Paolucci, and M. Wagner: Towards Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective. In OWL-S Experiences and Future Developments Workshop at ESWC 2007, June 2007, Innsbruck, Austria
[21]Kunal Verma and Amit Sheth: Semantically Annotating a Web Service, in IEEE Internet Computing 11 (no. 2), March–April 2007, pp. 83–85.
[22] WSMO-Grounding. http://www.wsmo.org/TR/d24/d24.2/v0.1/20070427/
[23] Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[21] N. Kavantzas, "Web Services Choreography Description Language 1.0," editor's draft, 3 Apr. 2004, W3C; http://lists.w3.org/Archives/Public/www-archive/2004Apr/att-0004/cdl_v1-editors-apr03-2004-pdf.pdf
[22] Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-oriented computing: state of the art and research challenges. Computer 2007];40(11):38–45.

[23] D. Bonino, F. Corno, L. Farinetti, A. Bosca. Ontology Driven Semantic Search. WSEAS Transaction on Information Science and Application, Issue 6, Volume 1, December 2004, pp. 1597-1605
[24] D. Bonino, F. Corno, L. Farinetti, A. Ferrato. Multilingual Semantic Elaboration in the DOSE platform. SAC 2004, ACM Symposium on Applied Computing, March 14-17, 2004, Nicosia, Cyprus
[25] D. Bonino, F. Corno, L. Farinetti. DOSE: a Distributed Open Semantic Elaboration Platform. ICTAI 2003, The 15th IEEE International Conference on Tools with Artificial Intelligence, November 3-5, 2003, Sacramento, California
[26] D. Bonino, F. Corno, L. Farinetti. Semantic annotation and search at the document substructure level. Poster at ISWC2003 - 2nd International Semantic Web Conference, Florida (USA), October 2003.
[27] EASY WSDL. Home Page. http://easywsdl.ow2.org/index.html
[28] XML2OWL. Home Page. http://xml2owl.sourceforge.net/index.php?input=about
[29] OWL2XSD. Home Page.
[30] Bouras, A.; Gouvas, P.; Kourtesis, D.; Mentzas, G.: Semantic Integration Of Business Applications Across Collaborative Value Networks. Springer: Boston 2007
[31] ITA. ICT for Logistic Work Group, http://sigma.ita.es/elogistica/index.php/en
[32] ITA. OpenBIP. http://sigma.ita.es/openBIP/

Improving Device-Aware Web Services and their Mobile Clients

Guadalupe Ortiz[1] Alfonso García de Prado

Quercus Software Engineering Group University of Cádiz, Spain.
gobellot@unex.es alfonso.garciaprado@mail.uca.es

Abstract[2]. Mobile devices have become an essential element in our daily lives and Web services have grown extremely important when offering services through the Internet. However, current Web services are very inflexible as regards their invocation from different types of device, especially when being invoked from mobile devices. In this paper, we summarize an approach for the creation of flexible Web services which can be invoked transparently from different types of device and which return subsequent responses, as well as providing the client's adaptation as a result of the particular device model characteristics and end user preferences in a completely decoupled way.

Keywords: Aspect-Oriented Software Development, Mobile Devices, Web Service, Model-Driven Development.

1 Introduction

Mobile devices have acquired great prominence over the last years; the great amount of devices and their continuous use clearly illustrate the importance of access to mobile services [1]. From the service-side point of view, Web service developers have mainly focused on developing services which are designed to be accessible from desktop or laptop computers, creating a void in the sphere of their access from mobile clients. In order to meet this requirement we have to bear in mind the type of device from which the service is going to be invoked. In this regard, developed clients will vary widely depending on the target device. Furthermore, not only screen size, but also runtime capacity needs to be considered [2]. On the other hand, from the client-side point of view, once we develop a client for a specific type of mobile device, we also have to account for the large existing differences across the whole assortment of mobile devices in the market. Besides, clients should also consider final user preferences: most mobile device screen settings can be personalized based on end-user preferences.

[1] Currently affiliated to University of Cádiz.

[2] This extended abstract summarizes the paper entitled *Improving Device-Aware Web Services and their Mobile Clients through an Aspect-Oriented, Model-Driven Approach* published at the Information and Software Technology Journal, Vol. 52, Issue 10, in 2010, p.1080-1093, which can be found at http://www.sciencedirect.com/science/article/pii/S0950584910000807

In this paper we summarize the solution presented in [3] for the creation of services which can be invoked from different types of device, particularly to adapt them to mobile clients, providing each one with the appropriate response. The solution is based on the transparent provision of information from the invoking device through the header of the sent SOAP message. The service will deliver one piece of information or another depending on the information it receives. In this approach Aspect-Oriented Programming has been used to avoid scattered and tangled code [4, 5] across the system due to the device adaptation. Furthermore, to save developers the need of learning an aspect-oriented language and to increase the implemented code's quality and reliability, device-related code has been automatically generated thanks to the use of a model-driven approach. Even more so, in regards with client-side applications, aspects allow us a modularized non-intrusive adaptation of the latter to the specific device characteristics in which it is going to be deployed as well as a dynamic adaptation to final user preferences. Aspects' performance evaluation has shown that their inclusion does not impact negatively on system performance.

Thus, the approach provides us with the possibility of following a model-driven development of mobile-aware Web services in an integrated platform: everything has been integrated in Eclipse, so that the full development process can be easily followed from initial platform-independent model to final implementation code.

Besides, the approach is perfectly extensible to different types of result the service has to return (not only mobile phone versus computer). Both platform-independent and platform-specific models are easily extensible for this purpose with the only addition of a stereotype for each new type of result to be returned. The new aspect required for the modularized and non-intrusive addition of code will be automatically generated by model-to-model and model-to-text transformations.

Furthermore, the use of aspects provides additional benefits: a last minute change of requirements would only imply the addition or deletion of an aspect, without the need to regenerate the full system.

Acknowledgments. The first author acknowledges the support from *Ministerio de Ciencia e Innovación* (TIN2008-02985) and from *Fondo Europeo de Desarrollo Regional* (FEDER) as well as research grant *José Castillejo*.

References

1. Pedersen P.E., Ling. R. Modifying adoption research for mobile Internet service adoption: cross-disciplinary interactions. Proc. Hawaii Int. Conf. on System Sciences, Hawai, 2003.
2. Nedos A., Kulpreet S., Clarke S. Mobile Ad Hoc Services: Semantic Service discovery in Mobile Ad Hoc Networks. P. Int. C. on Service-Oriented Computing, Chicago (USA), 2006
3. Ortiz, G. Garcia de Prado, A. Improving Device-Aware Web Services and their Mobile Clients through an Aspect-Oriented, Model-Driven Approach. Information and Software Technology Journal, Vol. 52, Issue 10, in 2010, p.1080-1093.
4. Kiczales, G. Aspect-Oriented Programming, ECOOP'97 Conference proceedings, LNCS 1241, June 1997.
5. Ortiz, G. Hernández, J. Aspect-Oriented Techniques for Web Services: A Model-Driven Approach. International Journal of Business Process Integration and Management. Vol. 2 Issue. 2, 2007.

Mapa de Procesos de la Dirección de Aplicaciones de Endesa Servicios

Francisco José Orgaz Lora[1], José Martínez Benavides[2]

[1] Dirección de Aplicaciones, Endesa Servicios, www.endesa.es
[2] Sopra Group, www.sopragroup.es

Abstract. Con la creación de la Dirección de Aplicaciones de Endesa Servicios y la definición de sus retos estratégicos, se decidió unificar la operativa de las tres Subdirecciones de Sistemas que englobaba para el desarrollo y mantenimiento de los Sistemas Técnicos, Comerciales y de Gestión. Por esto, se decidió invertir en la realización del Proyecto de Elaboración e Implantación del Mapa de Procesos de la Dirección de Aplicaciones, que permitió reducir los costes y mejorar la operativa de la Organización. Un factor clave de éxito fue la Gestión del Cambio, tanto en la propia Organización como con Clientes y Proveedores. La implantación del Mapa de Procesos sirvió como punto de partida para una nueva etapa actualmente en curso: la Globalización.

Keywords. mapa de procesos, Endesa servicios, MPRO

1 Situación de Partida y Enfoque de la Solución

Este artículo es un extracto del publicado en la *Revista de Procesos y Métricas de AEMES*, volumen 7, número 3, Septiembre - Diciembre de 2010 (ISSN 1698-2029), titulado "*Mapa de procesos de la dirección de aplicaciones de Endesa servicios*", de los mismos autores.

A principios de 2007, tuvo lugar en Endesa Servicios la creación de la denominada Dirección de Aplicaciones, que engloba a tres Subdirecciones de Línea responsables del desarrollo y mantenimiento de aplicaciones y orientadas al negocio: Subdirección de Sistemas Técnicos, Subdirección de Sistemas Comerciales y Subdirección de Sistemas de Gestión. Cada una de ellas realiza sus actividades según su propio modelo operativo. La Dirección asumió como reto superar los inconvenientes que se derivaban de esta situación.

Se ha elaborado un modelo único y homogéneo para la Dirección de Aplicaciones, incluyendo en él todos los procedimientos y plantillas asociados. Es una representación de sus actividades, constituyendo un Mapa de Procesos de la misma. Consigue integrar las interfaces entre los distintos modelos de gestión de proyectos e incidencias: gestión física, económica y documental. Orienta la Organización a la Mejora Continua y está alineado con estándares y normativas internacionales (CMMi, Ley Sarbanes-Oxley, Norma UNE-166002 en proyectos de I+D+i). Está descrito mediante una extensión de UML definida por Endesa para facilitar su entendimiento.

Se ha conseguido un modelo útil, orientado fundamentalmente a ayudar al equipo de la Dirección de Aplicaciones a realizar su trabajo diario.

2 Análisis de Viabilidad

Costes. Hay que considerer el coste de los trabajos de una consultora externa seleccionada por licitación (ascienden a 350.000€) y la dedicación de los integrantes de la Dirección de Aplicaciones, que fueron participantes claves durante todo el proyecto.

Beneficios. Se ha conseguido un ahorro del 9,2% en el presupuesto de 2008 de la Dirección de Aplicaciones respecto al año anterior debido a la mejora de la eficiencia. Se unen beneficios intangibles: productividad, flexibilidad, unificación de criterios de calidad, intercambio de conocimiento entre áreas y fábricas, reutilización del software, estandarización, mayor satisfacción y capitalización del conocimiento.

3 Gestión del Cambio en la Organización, con Clientes y con Proveedores

La Organización. El factor clave de éxito fue planificar un conjunto de acciones enmarcadas en cuatro líneas de actuación: Involucrar, Comunicar, Formar y Apoyar. Esto permitió mitigar el alto impacto del cambio.

Clientes y Proveedores. También fueron destinatarios de la comunicación. Se realizaron presentaciones explicativas y se ofreció soporte metodológico. La implantación del nuevo modelo les aportó beneficios, favoreciendo así su aceptación.

4 Próximas Etapas

Mejora Continua. El Modelo ha evolucionado a través del proceso de *Transformación del Proceso*, agilizando y automatizando procesos, enriqueciéndolos con las mejores prácticas y fortaleciendo el Aseguramiento de la Calidad.

Modelo de Software Factories. Se estableció un modelo de Software Factories por cada Área que nos ha permitido orientar los recursos a tareas de mayor valor añadido, aprovechar economías de escala y reducir y contener las tarifas.

Próximo horizonte: Globalización (Globaliza IT). Su meta es evolucionar el modelo de gobierno de TIC para crear más valor para los clientes y los accionistas, establecer Centros de Competencia, acelerar la consolidación y deslocalización de actividades de back-office, ya iniciadas a nivel nacional o regional, capitalizando nuestra escala y presencia global para la optimización de infraestructuras y operaciones y producción de software, así como impulsar, retomar o iniciar la globalización de componentes de los aplicativos que la lógica de los negocios o las posibilidades de la tecnología así lo recomienden.

Índice de autores/as

Patrocinadores:

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA E INNOVACIÓN

InterSystems

Deputación da Coruña

Tecnocom

enxenio

citic

Banco Pastor