MigraSOA: Migrando aplicaciones web legadas hacia arquitecturas orientadas a servicios (SOA)*

Encarna Sosa-Sánchez, Pedro J. Clemente, Álvaro Prieto, José M. Conejero, and Roberto Rodríguez-Echeverría

Universidad de Extremadura. Quercus Software Engineering Group,
{esosa, pjclemente, aeprieto, chemacm, rre}@unex.es,
http://quercusseg.unex.es

Resumen La migración de aplicaciones legadas hacia arquitecturas orientadas a servicios (SOA) es un proceso relativamente habitual en la actualidad, ya que las características de flexibilidad arquitectónica que ofrece SOA permiten adaptar fácilmente las aplicaciones a los nuevos requisitos marcados por las empresas. Sin embargo, el desarrollo de esta migración hacia estas nuevas arquitecturas software se lleva a cabo normalmente de forma manual, siendo este un mecanismo tedioso y propenso a errores. MigraSOA es una propuesta de migración de aplicaciones web legadas (LWA) hacia SOA que utiliza técnicas de Desarrollo de Software Dirigido por Modelos (MDD) para abordar la complejidad de las tecnologías subyacentes (servicios web, definición de procesos de negocio o plataformas para procesos de negocio ejecutables). En este trabajo, además de presentar MigraSOA de una forma global, nos centraremos en los aspectos de alineación de los procesos de negocio definidos por la empresa con los servicios web subyacentes en la aplicación legada y en cómo extender los modelos BPMN para conseguir la sincronización entre ellos y los servicios disponibles.

Keywords: migración de aplicaciones web, servicios web, procesos de negocio, arquitecturas orientadas a servicios

1 Introducción

Las empresas en general están sometidas a constantes cambios en sus procesos de negocio, lo cual requiere un esfuerzo importante para mantener sincronizados los procesos de negocio y su infraestructura tecnológica [14]. No obstante, las empresas deben rentabilizar las inversiones que en su momento realizaron en esta infraestructura tecnológica y además implementar la funcionalidad necesaria para soportar dichos cambios en sus procesos de negocio. En muchos casos

* This work was funded by Ministerio de Economía y Competitividad (Spain) - Project TIN2015-6957-R; the Junta de Extremadura, Consejería de Economía e Infraestructuras under grant GR15098 and by the European Regional Development Fund (ERDF).

esta infraestructura tecnológica está soportada por aplicaciones web que hoy en día pueden considerarse obsoletas y que deben ser rediseñadas para ofrecer arquitecturas más flexibles y que puedan responder a este continuo cambio de requisitos.

Además, las compañías deben definir sus procesos de negocio que posteriormente serán implementados mediante servicios ad hoc o adquiridos de terceros, convirtiendo el desarrollo de software en un auténtico proceso de composición de servicios interoperables [6]. En este contexto, las arquitecturas orientadas a servicios (SOA) han supuesto una auténtica revolución para el desarrollo de sistemas complejos, que integren servicios desarrollados bajo múltiples plataformas y que deben coordinarse para satisfacer los procesos de negocio de las compañías [6, 11].

Llegados a este punto, por una parte encontramos aplicaciones web legadas susceptibles de ser modernizadas y, por otra, una filosofía de desarrollo de software, SOA, que facilita la construcción de aplicaciones interoperables, extensibles, fiables, disponibles, escalables y adaptables [8].

La modernización de aplicaciones legadas ha tendido hacia la exposición de sus funcionalidades como servicios web [2, 3, 5, 7, 10, 17], sin embargo, en la mayoría de estas modernizaciones no se aborda la sincronización de los procesos de negocio de las empresas con estos servicios. En [9] se aboga por una propuesta dirigida por modelos para realizar la migración de aplicaciones web legadas hacia *Rich Internet Applications* (RIAs). Este trabajo ofrece un mecanismo de ingeniería inversa para abordar el análisis del código de la aplicación web legada (LWA) y obtener un modelo MVC (Modelo Vista Controlador) que la representa conceptualmente. Ello habilita la utilización de técnicas de desarrollo dirigido por modelos para modernizar la LWA hacia una nueva arquitectura, ya que a partir de la información disponible en dicho modelo, podremos abordar las transformaciones modelo a modelo y modelo a código necesarias para avanzar en la nueva versión de la LWA, alineada con los procesos de negocio de la compañía.

Así, a partir de [9], en [12, 13] se han presentado los aspectos esenciales de MigraSOA, un marco para la modernización de aplicaciones web legadas hacia SOA. La propuesta MigraSOA además de ofrecer un mecanismo dirigido por modelos para representar conceptualmente la aplicación orientada a servicios y generar los servicios web que actuarán como envoltorios (*wrappers*) sobre la LWA, también aborda la alineación de procesos de negocio (definidos por las compañías) con los servicios web legados (obtenidos a partir las aplicaciones web migradas). Las principales contribuciones de este trabajo son las siguientes:

- Definición del marco de modernización de aplicaciones web legadas denominado MigraSOA.
- Alineación de los procesos de negocio definidos en BPMN con los servicios web subyacentes que encapsulan la funcionalidad de la LWA.
- Extensión de los procesos de negocio definidos en BPMN e incorporación automática del código necesario para convertirlos en BPMN ejecutables, los cuales podrán interactuar con los servicios web obtenidos en el proceso MigraSOA.

La estructura del artículo se resume a continuación. En la sección 2 se presenta de forma general la propuesta MigraSOA. En la sección 3 se describe la definición de procesos de negocio y el proceso de alineación de los mismos con los servicios web de la LWA. Posteriormente, en la sección 4 se describe el proceso de extensión de los modelos BPMN para convertirlos en BPMN ejecutables. En la sección 5 se presentan las conclusiones obtenidas y los trabajos futuros.

2 MigraSOA: Migración semi-automática hacia SOA

Con el objetivo de hacer que este trabajo pueda ser presentado en su contexto, en esta sección se presenta una breve descripción de las cuatro fases principales que han sido definidas en la propuesta de migración LWA hacia SOA [12]. Estos pasos pueden observarse en la Figura 1 y se resumen en:

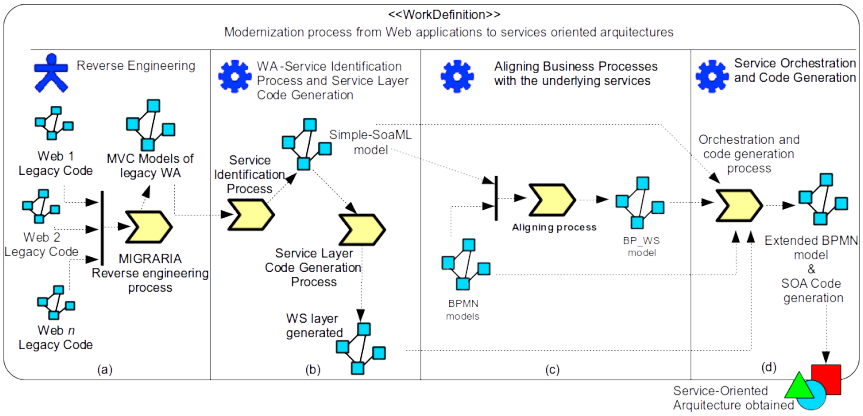

Fig. 1. Esquema de la migración de aplicaciones web legadas hacia SOA

a) Ingeniería inversa. El punto de partida de esta propuesta es una fase (identificada como *a* en la Figura 1) en la que, mediante el análisis estático de los artefactos software de una LWA, implementado con técnicas y herramientas de reingeniería, se obtiene su representación conceptual conforme al metamodelo MIGRARIA-MVC, desarrollado en el proyecto MIGRARIA [9].
b) Identificación de servicios y generación de código de la capa de servicios. En esta fase (*b* en la Figura 1) se refina el modelo conceptual etiquetando los servicios localizados en base a una búsqueda de determinados patrones (ejemplo de servicios podrían ser: buscar un artículo por su ID, añadir un artículo, comprobar el estado de un artículo, etc.). Este paso es importante para el proyecto completo, ya que una apropiada identificación de los servicios permite generar adecuadamente la capa de servicios. El modelo

obtenido en este paso (Simple-SoaML model) es conforme con un metamodelo denominado Simple-SoaML [12]. A partir de este modelo, se ha definido una transformación modelo a texto que permite generar de forma automática el código correspondiente a la capa de servicios subyacentes, la cual facilita, mediante servicios web tipo SOAP, interactuar con la funcionalidad de la LWA [12].

c) Alineación de procesos de negocio con los servicios subyacentes. El objetivo de esta fase (*c* en la Figura 1) es alinear los procesos de negocio definidos por la compañía con la capa de servicios generada en la fase anterior, para ello, se implementa un proceso de alineación que toma como entrada, por una parte, los modelos de procesos de negocio de la empresa, definidos en BPMN y, por otro lado, el modelo Simple-SoaML (obtenido en la fase *b*). El resultado de este proceso de alineación es un modelo denominado BP-WS que relaciona las tareas de los procesos de negocio con los servicios web generados en la fase *b*.
d) Orquestación de servicios y generación de código. Los procesos de negocios definidos por la compañía describen cómo los servicios subyacentes deben coordinarse. Así, esta orquestación de servicios se obtiene mediante un proceso de tejido (*weaving*) de modelos utilizando como fuente los modelos BPMN, el modelo Simple-SoaML y el modelo BP-WS obtenido en la fase *c*, obteniéndose un nuevo modelo denominado Extended BPMN. Este modelo corresponde a una extensión de los modelos originales BPMN incluyendo las referencias concretas a los servicios alineados. En nuestro caso, estos nuevos modelos BPMN son ejecutables mediante Apache Activiti [1].

En este trabajo se toma como punto de partida las dos primeras fases de MigraSOA, por lo que se asume que el proceso de identificación de servicios desde la LWA ha sido completado [12]. A continuación, nos centraremos en las fases *c* y *d* de la Figura 1, en las que: i) se define un proceso de alineación de servicios utilizando un algoritmo semántico y ii) se realiza un proceso de orquestación de servicios y generación de código que permite inyectar información adicional en los modelos BPMN.

3 Alineación de los procesos de negocio

En esta sección se aborda la alineación de los procesos de negocio con los servicios web llevada a cabo en la fase *c*. La compañía debe definir en BPMN los procesos de negocio. Además, estos procesos de negocio deben definirse siguiendo las guías para la descripción de procesos de negocio operacionales definidas en [16], de forma que se aporte suficiente información sobre los mensajes entre tareas, incluyendo la información de parámetros de entrada y salida de las distintas tareas. En la figura 2 puede observarse un ejemplo de proceso de negocio sobre un sistema denominado CRS (*Conference Review System*).

La alineación automática de los procesos de negocio con los servicios web identificados se lleva a cabo utilizando un algoritmo semántico [13]. Este

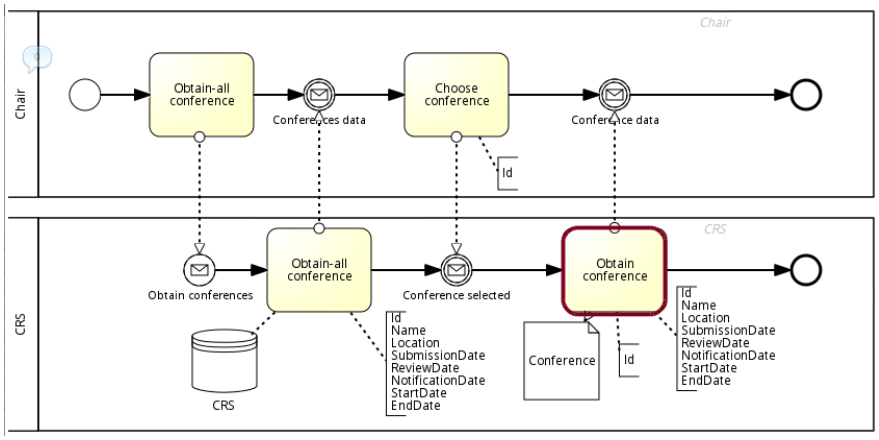

Fig. 2. Diagrama BPMN que representa el proceso de negocio *GetConference*

algoritmo ha sido adaptado para manejar la complejidad relacionada con la identificación de las tareas de los procesos de negocio y el análisis de los servicios web disponibles. Como ejemplo de identificación de las tareas definidas en los procesos de negocio, en la figura 2 se puede observar la tarea *Obtain conference* marcada con una línea más gruesa, sobre la cual se observan los atributos relacionados con la información general de una conferencia (*Id*, *Name*, *Location*, *SubmissionDate*, *ReviewDate*, *NotificationDate*, *StartDate* y *EndDate*). Para llevar a cabo la alineación se utiliza como base semántica un diccionario que representa el dominio de aplicación y se apoya fundamentalmente en la definición de un conjunto de términos y sus sinónimos para identificar posibles alineaciones semánticas entre las tareas de los procesos de negocio y los servicios web identificados en la LWA.

En Algoritmo 1 se describe el pseudocódigo correspondiente a la adaptación del algoritmo semántico de Wang-Ali [15], que originalmente fue propuesto para obtener una medida de similitud entre ontologías, de forma que cuanto más cercana a 1 era dicha medida las ontologías comparadas no tenían nada en común y cuanto más cercana a 0, más similares eran. Esta adaptación se ha realizado en dos partes: por una parte, el algoritmo original compara ontologías completas y en la versión adaptada se comparan dos conjuntos de términos: a) tareas y sus sinónimos con b) servicios y sus sinónimos (el conjunto de sinónimos de cada tarea o servicio son los términos relacionados según el diccionario semántico definido). Por otra parte, las etiquetas en las tareas y en los identificadores de los servicios están formadas por palabras compuestas, incluyendo una acción y un objeto, de modo que podemos identificar cada palabra y aplicar un peso configurable a cada una de ellas en la aplicación del algoritmo. Como ejemplo, la tarea *Obtain conference* mostrada en la figura 2 puede dividirse en dos palabras: *Obtain* (acción) y *conference* (objeto).

En el Algoritmo 1 se define $\mathcal{T}$ como el conjunto de tareas de procesos de negocio y $\mathcal{S}$ como el conjunto de servicios. De tal forma que, para cada tarea (llamada T) en $\mathcal{T}$, se comprueba cada servicio (llamado S) en $\mathcal{S}$ y se calcula un valor de similitud entre cada tarea y cada servicio. Este valor estará en el rango [0-1], de modo que cuanto más cercano sea el valor a 0, más similares serán los términos comparados. Para diferenciar acción y objeto en cada término, nombraremos *T1* a la acción de la tarea T y *T2* al objeto de la tarea T. Lo mismo aplicamos en los servicios: *S1* será la acción del servicio S y *S2* será el objeto del mismo servicio. Por último, se aplican pesos a la acción y objeto que se denominan *weighta* y *weighto* respectivamente.

Algoritmo 1 Algoritmo semántico para el proceso de alineación [12]

```
Wang-Ali-extended algorithm (T, S, weighta, weighto, Wang-Ali-SimMatr[][])
begin
for each T ∈ T    //for each task
      T1 = action of T
      T2 = object of T
      Syn_T1 = set of synonyms of T1
      Syn_T2 = set of synonyms of T2
for each S ∈ S   //for each service
            S1 = action of S
            S2 = object of S
            Syn_S1 = set of synonyms of S1
            Syn_S2 = set of synonyms of S2
            v1 = Syn_T1 - Syn_S1 / Syn_T1
                 //v1: semantic diff. between actions
            v2 = Syn_T2 - Syn_S2 / Syn_T2
                 //v2: semantic diff. between objects
            Wang-Ali-SimMatr[T][S] = v1 * weighta +
                                     v2 *  weighto
      end for
end for
end
```

El resultado obtenido de este algoritmo corresponde a una matriz *Wang-Ali-SimMatr* de t filas (número de términos en el conjunto de tareas de procesos de negocio) y s columnas (número de términos en el conjunto de servicios). La i-ésima fila y j-ésima columna de la matriz contiene el valor $Wang\text{-}Ali\text{-}SimMatr_{ij} = Wang\text{-}Ali\text{-}SimMatr\,[t_i]\,[s_j]$, que representa el valor de similitud entre cada tarea definida en los procesos de negocio y cada servicio definido en la capa de servicios identificada en la fase b [12]. Como hemos indicado previamente, para cada $Wang\text{-}Ali\text{-}SimMatr[t_i][s_j]$ se obtiene un valor en el rango entre [0-1] que permite identificar la posible alineación entre una tarea y un servicio. Se ha establecido que sólo aquellas comparaciones que superen un umbral de 0.8 [13] (este umbral puede ser modificado por el ingeniero software) se consideran

alineaciones adecuadas. Téngase en cuenta que el cálculo del umbral ha sido normalizado para representar valores comparables ascendentes: $1 - Wang\text{-}Ali\text{-}SimMatr_{ij}$.

El resultado obtenido en esta fase *c* incluye: i) tareas de procesos de negocio alineadas con los servicios subyacentes y ii) tareas de procesos de negocio que no han podido ser alineadas con ningún servicio. Ambas situaciones se recogen sobre un modelo denominado BP-WS (conforme con el metamodelo del mismo nombre, BP-WS) donde se modelan las relaciones de alineación obtenidas.

4 Generación de la orquestación de los servicios mediante la extensión de los modelos BPMN

Los procesos de negocio describen cómo distintos servicios deben estar orquestados para alcanzar un determinado objetivo. En esta fase *d* partimos de los modelos BPMN, el modelo Simple-SOAML y el modelo que representa las tareas de procesos de negocio alineadas y los servicios web que deben implementarlas (modelo BP-WS). A partir de estas entradas se extienden los procesos de negocio (BPMN) con la información específica sobre la invocación a los servicios indentificados, obteniéndose así una nueva versión extendida de los modelos BPMN, los cuales serán ejecutables. En otras palabras, ya que en la fase *c* de MigraSOA se obtiene un modelo que identifica la alineación de tareas en los procesos de negocio y servicios, es necesario llevar esta información a los modelos concretos que las plataformas de ejecución de procesos de negocio requieren. En consecuencia, este proceso es específico de la plataforma concreta de ejecución de los procesos de negocio, en nuestro caso Apache Activiti [1].

La generación de código llevada a cabo permite: i) inyectar en los archivos .bpmn el código necesario para invocar los correspondientes servicios y ii) generar las clases Java que sirven de *wrapper* para invocar el servicio web requerido.

Por una parte, se debe localizar en el archivo .bpmn la información sobre la correspondiente tarea e inyectar la información sobre el servicio que debe ser invocado, incluyendo la dirección URL donde se localiza el WSDL del servicio web, el nombre de la operación que se va a invocar y el mapeo de parámetros de entrada y salida. Además, se debe definir la ubicación de una clase Java que será la encargada de llevar a cabo la invocación desde el motor de ejecución. Esta clase Java debe implementar la interface *org.activiti.engine.delegate.JavaDelegate*.

En este caso, la implementación de dicha interface se realiza utilizando Apache CXF [4], una implementación de JAX-WS [1], que facilita la construcción de invocaciones sobre servicios web. Cabe destacar que la información desde el motor de ejecución hacia la clase Java definida es un parámetro de tipo *DelegateExecution execution*, a partir del cual podemos obtener la URL del WSDL, el nombre de la operación y la lista de parámetros de entrada y salida. En consecuencia, desde la implementación de esta clase podemos realizar la invocación al servicio.

[1] https://jcp.org/en/jsr/detail?id=224

Por lo tanto, la implementación concreta de esta interface permite: i) obtener una instancia dinámica del WS a partir de *JaxWsDynamicClientFactory* [4]; ii) crear un cliente para el WSDL del WS a invocar; iii) realizar la invocación pasando como argumentos la operación y un vector con los datos de entrada; y iv) obtener el resultado almacenado en un vector de datos de salida.

En la Figura 3 podemos observar el código XML necesario para extender una tarea en un proceso BPMN. Como puede observarse, será necesario vincular la tarea BPMN alineada con la clase que implementa la invocación del servicio Web (previamente descrita) y para ello se etiquetan los siguientes parámetros de la tarea: i) *TaskType* que identifica el tipo de tarea, específicamente *serviceTask*; ii) *activiti:class,* clase definida previamente que implementa la interface *org.activiti.engine.delegate.JavaDelegate* denominada *classes.WSDelegate*; iii) parámetros configurables, entre los que se añaden: *wsdl, operation, alignedParamsIn, alignedParamsOut y notAlignedParamsOut.* Los parámetros *alignedParamIn, alignedParamOut y notAlignedParamOut* forman parte del resultado obtenido en la fase anterior de MigraSOA (fase *c*), donde se alinean las tareas de los procesos de negocio con los servicios web que deben implementarlas.

```
<serviceTask id="serviceTask1" name="Name␣of␣the␣Service␣Tasks"
        activiti:class="classes.WSDelegate">
        <extensionElements>
        <activiti:field name="wsdl">
                <activiti:expression>URL where the WSDL web service
            is available</activiti:expression>
        </activiti:field>
        <activiti:field name="operation">
                <activiti:expression>operationName
            </activiti:expression>
        </activiti:field>
        <activiti:field name="alignedParamsIn">
                <activiti:expression>${paramIn1}
            </activiti:expression>
        </activiti:field>
        <activiti:field name="alignedParamsOut">
                <activiti:expression>${paramOut1}
            </activiti:expression>
        </activiti:field>
        <activiti:field name="notAlignedParamsOut">
                <activiti:expression>${notAlignedParamOut1}
            </activiti:expression>
        </activiti:field>
        </extensionElements>
</serviceTask>
```

Fig. 3. Ejemplo de código XML asociado a una tarea BPMN de tipo servicio

Como resultado en esta fase *d*, los modelos BPMN han sido extendidos mediante la inyección del código necesario para invocar adecuadamente los servicios que han sido alineados. En definitiva, se obtienen los procesos BPMN ejecutables adaptados al motor de procesos de negocio Apache Activiti. Cabe destacar que en los procesos BPMN pueden existir tareas que no han sido alineadas con servicios web, las cuales deben ser revisadas manualmente ya que podrían requerir la implementanción adicional de algún servicio web que no está soportado por la LWA.

5 Conclusiones

En este trabajo se aborda, por una parte, la migración desde LWA hacia arquitecturas orientadas a servicios la cual puede realizarse utilizando técnicas de desarrollo de software dirigido por modelos que permiten manejar la complejidad de las plataformas tecnológicas actuales. Por otra parte, se permite a las empresas definir sus procesos de negocio, los cuales serán alineados con los servicios web ofrecios por la LWA, facilitando de este modo la orquestación de los servicios en la arquitectura SOA. Además, se ha implementado un proceso de generación de código que permite extender la definición original de los procesos de negocios definidos en BPMN para inyectar el código necesario que permite la invocación de los servicios web alineados.

Actualmente se está trabajando en otros casos de estudio que incluyan procesos de negocios más complejos y nuevos generadores de código hacia otras plataformas de ejecución de procesos de negocio tales como Bonita Soft.

Referencias

1. Activiti, A.: Apache Activiti, a Business Process Management (BPM) platform. http://www.activiti.org (2015), `http://www.activiti.org`
2. Almonaies, A., Cordy, J.R., Dean, T.R.: Legacy system evolution towards service-oriented architecture. In: International Workshop on SOA Migration and Evolution. pp. 53–62 (2010)
3. Almonaies, A.A., Alalfi, M.H., Cordy, J.R., Dean, T.R.: A framework for migrating web applications to web services. In: Daniel, F., Dolog, P., Li, Q. (eds.) Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7977, pp. 384–399. Springer (2013), `http://dx.doi.org/10.1007/978-3-642-39200-9`
4. Apache CXF: CXF User's Guide. Tech. rep. (2015), `http://cxf.apache.org/docs/`
5. Canfora, G., Fasolino, A.R., Frattolillo, G., Tramontana, P.: A wrapping approach for migrating legacy system interactive functionalities to Service Oriented Architectures. The Journal of Systems and Software 81(4), 463–480 (Apr 2008)
6. Davis, J.: Open Source Soa. Manning Publications Co., Greenwich, CT, USA, 1st edn. (2009)
7. Liu, Y., Wang, Q., Zhuang, M., Zhu, Y.: Reengineering legacy systems with restful web service. In: Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. pp. 785–790. IEEE (2008)

8. O'Brien, Liam; Bass, L., Merson, P.F.: Quality attributes and service-oriented architectures. Software Engineering Institute. Paper 449. http://repository.cmu.edu/sei/449/ (2005)
9. Rodríguez-Echeverría, R., Conejero, J.M., Linaje, M., Preciado, J.C., Sánchez-Figueroa, F.: Re-engineering legacy Web applications into Rich Internet Applications. In: 10th International Conference on Web Engineering. vol. 6189, pp. 189–203. LNCS, Springer (2010)
10. Sneed, H.M.: Integrating legacy software into a service oriented architecture. In: CSMR. pp. 3–14. IEEE Computer Society (2006)
11. SOA-Manifesto: http://www.soa-manifesto.org (2011)
12. Sosa, E., Clemente, P.J., Conejero, J.M., Rodríguez-Echeverría, R.: A model-driven process to modernize legacy web applications based on service oriented architectures. In: WSE. pp. 61–70 (2013)
13. Sosa, E., Clemente, P.J., Sánchez-Cabrera, M., Conejero, J.M., Rodríguez-Echeverría, R., Sanchez-Figueroa, F.: Service discovery using a semantic algorithm in a soa modernization process from legacy web applications. In: IEEE Tenth World Congress On Services. SERVICES. IEEE Computer Society (2014)
14. Ullah, A., Lai, R.: A systematic review of business and information technology alignment. ACM Transactions on Management Information Systems (TMIS) 4(1), 4 (2013)
15. Wang, J.Z., Ali, F., Appaneravanda, R.: A web service for efficient ontology comparison. In: Proceedings of the IEEE International Conference on Web Services. pp. 843–844. ICWS '05, IEEE Computer Society, Washington, DC, USA (2005)
16. Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd Edition. Springer (2012), http://dx.doi.org/10.1007/978-3-642-28616-2
17. Zhang, B., Bao, L., Zhou, R., Hu, S., Chen, P.: A black-box strategy to migrate gui-based legacy systems to web services. In: Service-Oriented System Engineering, 2008. SOSE'08. IEEE International Symposium on. pp. 25–31. IEEE (2008)