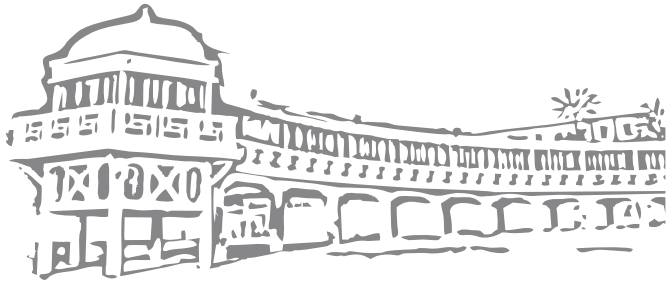

Jornadas Sistedes 2014

Cádiz, del 16 al 19 de septiembre

JISBD PROLE JCIS DC

X Jornadas de Ciencia e Ingeniería de Servicios

ACTAS

X Jornadas de Ciencia e Ingeniería de Servicios

Editores:

Félix García
Mercedes Ruiz
Elena Orta

Actas de las X Jornadas de Ciencia e Ingeniería de Servicios (JCIS)

Cádiz, 17 y 18 de septiembre 2014

Editores: Félix García, Mercedes Ruiz y Elena Orta

ISBN-10: 84-697-1153-9

ISBN-13: 978-84-697-1153-8

Prólogo

En el presente volumen se recogen los artículos seleccionados para su presentación en las X Jornadas de Ciencias e Ingeniería de Servicios (JCIS 2014), celebrado el 17 y 18 de Septiembre de 2014 en Cádiz, España.

Las Jornadas de Ciencia e Ingeniería de Servicios, desde su nacimiento 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), suponen un foro de referencia nacional para el intercambio de conocimiento de investigación y experiencia industrial en temas tan estratégicos para las organizaciones como la gestión de servicios y de procesos de negocio. El interés por tanto no sólo se centra en los nuevos avances científicos, sino también en las tecnologías existentes en torno a la computación orientada a servicios y los procesos de negocio, las nuevas prácticas de ingeniería de servicios y las lecciones aprendidas por medio de experiencias reales.

Con estos objetivos en mente, en la llamada a trabajos de las jornadas se solicitaron tres tipos de contribuciones: artículos de innovación y/o investigación, artículos ya publicados en revistas y congresos de especial relevancia y experiencias prácticas en el dominio de la empresa y/o administración, dentro del contexto de las siguientes temáticas generales de interés: Ingeniería de servicios; Gestión de los Servicios; Procesos de Negocio; Arquitectura SOA y patrones arquitecturales. Como resultado, en esta edición de las jornadas se recibieron 20 contribuciones, 16 de las cuales son artículos de innovación y/o investigación y 4 son artículos ya publicados en revistas o congresos de reconocido prestigio de acuerdo a las bases de la llamada de artículos de las jornadas. Las distintas contribuciones fueron revisadas por al menos tres miembros del comité de programa, recibiendo todas ellas una valoración positiva y por tanto siendo aceptadas para su presentación en las jornadas.

El programa de JCIS 2014 se ha organizado en torno a cuatro sesiones temáticas en las que se han distribuido las presentaciones de los trabajos a lo largo de dos días tratándose los siguientes temas: Composición de Servicios; Aplicaciones de Procesos y Servicios; Gestión de Recursos; Calidad y Mejora en Procesos y Servicios. Se incluye además una sesión especial en la que se imparte la conferencia invitada titulada “Being Digital: The Power of the Bits. Science, University and Industry perspective", trabajo elaborado por Carlos Fernandez, Jordi Guijarro y Jesús Bermejo, y que aportan una perspectiva de integración de los ámbitos académico e industrial. El programa de las jornadas JCIS cuenta además con los temas de gran interés abordados en las dos conferencias invitadas en el marco de las jornadas SISTEDES 2014, impartidas por los conferenciantes internacionales Jan Mendling: “Automatic Identification of Service Candidates from Business Process Models“; y por Antonia Bertolino: “Software Testing and/or Software Monitoring: Differences and Commonalities”.

Me gustaría agradecer a todos aquellos que con su dedicación y esfuerzo han contribuido a la organización de estas Jornadas y que hacen que sean un éxito año tras año. En primer lugar, a todos los autores de los artículos enviados a JCIS 2014, a Guadalupe Ortiz por su gran labor en la coordinación de trabajos ya publicados y a los miembros del Comité de Programa por su disponibilidad, dedicación y excelente trabajo a la hora de realizar sus revisiones. Agradecer además a nuestros colaboradores: ATI, Novática, y la Red Científico-Tecnológica

en Ciencias de los Servicios financiada por el Ministerio de Economía y Competitividad. Finalmente, agradecer a la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo del Software (SISTEDES) y a los miembros de la Universidad de Cádiz encargados de la organización de las jornadas. Es destacable el esfuerzo y el compromiso necesario en la organización de este tipo de eventos, en especial ante situaciones de dificultades económicas como las que nos encontramos, por lo que es de agradecer enormemente su trabajo para que todo sea un éxito.

Cádiz, Septiembre 2014

Félix O. García

Presidente del Comité científico

Prólogo de la organización

La décima edición de las Jornadas de Ingeniería y Ciencia de Servicios (JCIS) se celebra en 2014 en la ciudad de Cádiz, en el ámbito de las Jornadas Sistedes 2014. Como organizadores, agradecemos al Comité Permanente de dichas Jornadas la confianza depositada en nosotros y confiamos en que nuestro trabajo les permita a todos disfrutar de unas Jornadas enriquecedoras tanto desde el punto científico como personal.

Queremos agradecer a los miembros del comité científico de JCIS su disponibilidad y trabajo efectuado durante las revisiones y la confección del programa científico del evento, así como a los chairs de las diferentes sesiones regulares su atención a los detalles particulares de cada una de ellas. Es imprescindible manifestar la excelente labor del presidente del comité científico, Félix García, que, con su implicación en todos los detalles relacionados con las Jornadas, nos ha facilitado en gran medida nuestra tarea.

También es necesario agradecer a las entidades colaboradoras su contribución en la celebración del evento. En tiempos como los actuales resulta, si cabe, aún más importante este agradecimiento. Agradecemos a la Universidad de Cádiz, a la Escuela Superior de Ingeniería y al Departamento de Ingeniería Informática las facilidades que nos han dado para el uso de las instalaciones y recursos. Queremos reconocer la importante contribución de los grupos de investigación de Mejora del Proceso Software y Métodos Formales y UCASE de Ingeniería del Software, porque han contribuido con lo más preciado que tienen que son sus miembros. De igual manera, queremos agradecer al Excmo. Ayuntamiento de Cádiz y a su Delegación de Turismo su atenta colaboración y atención en la organización de algunas de las actividades sociales del programa de las Jornadas. De igual manera, agradecemos a las empresas Renfe e Iberia su colaboración a la hora de poner a disposición de los asistentes los descuentos para el desplazamiento a Cádiz.

Finalmente, nuestro agradecimiento más particular y especial debe ir dedicado a todos aquellos que han colaborado en los trabajos de organización de las Jornadas. Agradecemos profundamente el tiempo, las energías, la implicación, el compromiso y el trabajo de todos los que han participado en la preparación y celebración de las Jornadas.

Os deseamos a todos una feliz estancia en Cádiz y confiamos en que las Jornadas constituyan un excelente foro de intercambio de conocimientos, experiencias y reflexión.

Cádiz, septiembre de 2014

Mercedes Ruiz

Presidenta del Comité Organizador
de las Jornadas Sistedes 2014

Comité científico

PRESIDENTE DEL COMITÉ DE PROGRAMA

Félix Óscar García Rubio (Universidad de Castilla La Mancha)

COORDINADORA DE DIVULGACIÓN DE ARTÍCULOS YA PUBLICADOS

Guadalupe Ortíz Bellot (Universidad de Cádiz)

MIEMBROS DEL COMITÉ DE PROGRAMA

Andrea Delgado (InCo)
Antonio Ruiz-Cortés (U. de Sevilla)
Antonio Vallecillo (U. de Málaga)
Carlos Bobed (U. de Zaragoza)
Daniel González Morales (ACIISI)
Diego López (Telefónica I+D)
Fernando González Ladrón de Guevara (U. Politécnica Valencia)
Francisco Almeida Rodríguez (U. La Laguna)
Francisco Ruiz (U. de Castilla-La Mancha)
Francisco Javier Fabra (U. de Zaragoza)
Guadalupe Ortiz (U. de Cádiz)
Javier Troya (U. de Málaga)
Jesús Arias Fisteus (U. Carlos III de Madrid)
Jesús Bermejo (Schneider Electric)
Jesús Gorroñogoitia (Atos Research & Innovation)
Joan Pastor (U. Oberta de Catalunya)
Jordi Marco (U. Politécnica de Cataluña)
José Emilio Labra (U. de Oviedo)
José Hilario Canós (U. Politécnica Valencia)
Jose Manuel Gómez (iSOCO)
José Raúl Romero (U. de Córdoba)
Juan Hernández (U. de Extremadura)
Juan José Moreno Navarro (U. Politécnica Madrid)
Juan Manuel Murillo (U. Extremadura)
Juan Pavón (U. Complutense de Madrid)
Leire Bastida (Tecnalia)
Manuel Lama (U. Santiago de Compostela)
Manuel Resinas (U. de Sevilla)
Marcos López Sanz (U. Rey Juan Carlos)
María del Carmen Penadés (U. Politécnica Valencia)

María Valeria De Castro (U. Rey Juan Carlos)
María Ribera Sancho (U. Politécnica de Cataluña)
Marta Patiño (U. Politécnica de Madrid)
Martín Álvarez Espinar (W3C Spain)
Mercedes Ruiz (U. de Cádiz)
Óscar Corcho (U. Politécnica de Madrid)
Pedro Alvarez (U. de Zaragoza)
Pere Botella (U. Politécnica de Cataluña)
Rafael Corchuelo (U. de Sevilla)
Silvia Acuña (U. Autónoma de Madrid)
Vicente Pelechano (U. Politécnica de Valencia)
Víctor Ayllón (Novayre)
Victoria Torres (U. Politécnica de Valencia)

Comité de organización

PRESIDENTA DEL COMITÉ ORGANIZADOR

Mercedes Ruiz Carreira

MIEMBROS DEL COMITÉ ORGANIZADOR

Juan Boubeta Puig
Mª del Carmen de Castro Cabrera
Pedro Delgado Pérez
Juan Manuel Dodero Beardo
Antonio García Domínguez
Mª Teresa García Horcajadas
Lorena Gutiérrez Madroñal
Nuria Hurtado Rodríguez
José Luis Isla Montes
Inmaculada Medina Bulo
José Miguel Mota Macías
Manuel Palomo Duarte
Elena Orta Cuevas
Guadalupe Ortíz Bellot
Carlos Rioja del Río
Iván Ruiz Rube
Alberto Gabriel Salguero Hidalgo

Entidades colaboradoras

Índice de contenidos

Keynotes

Charla invitada

Composición de servicios

Chair: Guadalupe Ortiz

Aplicaciones de procesos y servicios

Chair: Javier Fabra

Gestión de recursos

Chair: Victoria Torres

Calidad y mejora en procesos y servicios

Chair: Manuel Resinas

Software Testing and/or Software Monitoring: Differences and Commonalities

Antonia Bertolino

ISTI-CNR
Via Moruzzi, 1
Pisa, Italy
antonia.bertolino@isti.cnr.it

Abstract. Validation is an essential part of the software life cycle. The actual functional and non-functional behaviour of an application needs to be checked against the expected or intended behaviour. Both testing and monitoring are widely used approaches for this purpose. More traditionally testing is considered as a technique for fault removal and forecasting during development. Monitoring is instead conceived for run-time observation of deployed software. Testing and monitoring approaches are usually contrasted as being, respectively, in-the-laboratory vs. in-the-field, and active vs. passive. In this talk I will overview concepts and techniques for software testing and monitoring, and will discuss how for modern pervasive and dynamic software systems the two approaches tend to converge in combined and synergic ways.

Automatic Identification of Service Candidates from Business Process Models

Henrik Leopold and Jan Mendling

Wirtschaftsuniversität Wien, Welthandelsplatz 1, 1020 Vienna, Austria
henrik.leopold@wu.ac.at, jan.mendling@wu.ac.at

Abstract. Although several approaches for service identification have been defined in research and practice, none of them considers the potential of fully automating the associated phases. As a result, users have to invest a substantial amount of manual work. In this paper, we address the problem of manual work in the context of service identification and present an approach for automatically deriving service candidates from business process models. Our approach combines different analysis techniques in a novel way in order to derive ranked lists of service candidates. The approach is meant to be a useful aid for enabling business and IT managers to quickly spot reuse potential in their company. We demonstrate the usefulness of our approach by reporting on the results from an evaluation with a process model collection from industry.

1 Introduction

Services-Oriented Architecture has been discussed for roughly a decade as a concept to increase the agility of a company in providing goods and services to external partners and organizing internal operations. In this context, a service can be understood as an action that is performed by an entity on behalf of another one, such that the capability of performing this action represents an asset [1]. The focus on services is supposed to improve business and IT alignment because it establishes principles like abstraction, autonomy, and reuse [2].

In the past, many approaches for identifying services from various input types have been defined [2]. Among others, these approaches leverage requirements documents [3,4], process models [5,6,7], and source code [8,9,10]. Particularly process models turn out to be a valuable source for identifying services since they are readily available in many companies. However, many of the approaches building on process models lack methodological detail and none of them considers the potential of fully automating the associated phases. The problem is that a manual approach does not scale up to the size of a whole company.

In this paper, we address the problem of manual work in the phases of service derivation and summarizes the results earlier presented in [11] and extend it with an additional identification technique. We consider the situation where an extensive set of hundreds of process models is available, which is often the case for many medium-sized and big companies [12]. Our contribution is an approach for the automatic derivation of service candidates, augmented with a set of

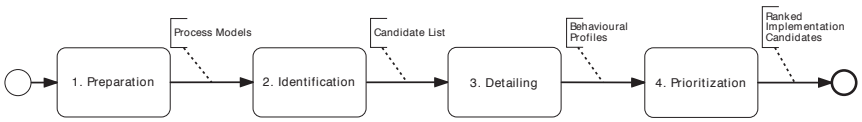

Fig. 1. The four phases of service derivation

metrics giving first clues about priorities. This approach is meant as a decision support tool for business and IT managers to quickly spot reuse potential in their company. In this way, the approach aims to speed up derivation drastically, and it can easily scale for involving large sets of process models of the whole company.

The paper is structured accordingly. Section 2 introduces the procedure of service derivation as it is summarized in related research. Section 3 introduces our approach for service identification. Section 4 presents the results of testing our prototypical implementation on a set of roughly 600 process models from practice. Section 5 discusses related work before Section 6 concludes the paper.

2 Background

In prior research, various approaches for identifying services have been defined [2]. Many of them distinguish between business services and software services. The concept of a business service puts more emphasis on the economic perspective, while a software service is more related to information technology. However, it has been shown that both service types have many commonalities. For both business and software services many authors consider the analysis and evaluation of business process models to be a central step [7,13,14]. Accordingly, we describe service derivation as a four phase approach involving preparation, identification, detailing, and prioritization, similar to the integrated approach of [2]. Figure 1 illustrates this approach.

The derivation of services usually starts with a *preparation phase*. In this phase, an information base for the service analysis is established. This information base may include different types of business documents such as enterprise architectures, business processes or organizational structures. In this paper, we assume that a collection of process models is already available. This is a viable assumption since big and medium-sized companies typically possess hundreds of process models [12]. The subsequent *identification phase* is concerned with identifying capabilities. In process models, these capabilities can be closely related to actions. If required, the available processes have to be further decomposed in order to arrive at a suitable level of detail. In the following *detailing phase*, the relationships and interactions between services are identified. This includes the detection of overlaps with existing services and the proper incorporation of new services into the existing SOA landscape. Finally, the *prioritization phase* is utilized to decide which services should be considered for implementation with which priority.

This four-phase process shows that the issue of scalability is hardly discussed. This is a significant problem when a service-oriented architecture is embraced as a company-wide concept. When starting from a process perspective, this means that dozens of processes have to be modeled. Often, it takes weeks to document only a single process. Even if a big number of process models already exists, it is hardly possible to inspect them manually in a systematic way. Against this background, it is striking that none of the previously defined service derivation approaches considers the potential of fully automating this task. In the following, we define an approach that assembles analysis techniques in an innovative way towards this end.

3 Automatic Service Identification and Detailing

This section describes our approach for the automatic identification and detailing of service candidates from a given set of process models as defined in [11]. It contains three main phases. First, we *annotate* each activity from the process model set with its action and business object. Second, we use different strategies to *identify* a list of service candidates from these activities. Third, we use behavioral profiles for *detailing* the service.

3.1 Annotation of Process Model Activities

In this phase, we annotate each activity with its *action(s)* and *business object(s)*. As an example, consider the activity *Notify customer*. It consists of the action *to notify* and the business object *customer*. To accomplish this annotation, we employ a the technique defined in [15]. It builds on the insight that activity labels follow regular structures, so-called labeling styles. By automatically recognizing these varying labeling styles, it reliably decomposes every activity label into action and business object. Note that this technique is also available for languages other than English [16].

3.2 Identification of Service Candidates

Building on the annotation with action and business object, we introduce four different approaches for identifying service candidates, and extend the approach of [11] with an Action-based Composite Service Identification technique.

Atomic Service Identification: The atomic service identification strategy focuses on single activities and is based on the notion that reoccurring activities are likely to represent relevant service candidates [17]. Consequently, the frequency of a particular activity throughout the model collection determines its potential of being a suitable service candidate. In order to capture these considerations, we introduce the activity frequency metric $F_{Activity}$, which determines the number of similar activities in a process model collection for a given activity. Note that the similarity is not based on the syntactical equivalence of the activity labels,

but on the similarity of the respective actions and business objects. Thus, for instance, our algorithm considers the activities *Notify Customer* and *Customer Notification* to be identical. As a result from this technique, we obtain a list of activity candidates ordered by their potential of representing suitable service candidates.

Composite Service Identification: The composite service identification approach abstracts from single activities and identifies composite services based on business object groups. To implement this strategy, we introduce the frequency metric F_{Object}, which determines the relevance of a group based on the occurrence of the business object among all activities of the considered model collection. As a result from this technique, we obtain a list of composite services candidates ordered by their relevance.

Inheritance Hierarchy Identification: The inheritance hierarchy identification approach is based on the considerations of the composite service identification strategy. However, it extends this approach by taking hierarchical relationships among business objects into account. This is motivated by the design principle of service cohesion [18], which refers to the degree of relatedness between the operations of a service. Assuming that activities with related business objects may also lead to related services, we identify business object hierarchies by decomposing the business objects. As an example, consider the business object *purchase order*, in which the word *purchase* specifies the main word *order*. In the context of the inheritance hierarchy identification, an activity with this business object is not only linked to other activities with the business object *purchase order*, but also to activities with the business object *order* as a sub concept. For implementing this strategy, we introduce the frequency metric $F_{Hierarchy}$, which defines the occurrence of the main word among all business objects. For each main word, we create a respective inheritance hierarchy tree and insert hierarchically related business objects on the according level. As a result of this technique, we obtain a list of hierarchy trees ordered by the frequency of their main word.

Action-based Composite Service Identification: The notion of a composite service can be also applied to the actions of a process model collection. This strategy is motivated by the observation that some actions are often associated with the same mechanism or procedure. As an example, consider the actions *print* or *contact*. To implement this strategy, we introduce the frequency metric F_{Action}, which determines the relevance of an action based on its number of occurrences in the model collection. As a result from this technique, we obtain a list of action-based composite service candidates ordered by their relevance.

3.3 Detailing of Service Candidates

Service detailing refers to the definition of the structure and behavior of a service or a set of services. To this end, we adopt an approach for mining action patterns, i.e., patterns that represent recurring behavior [19]. The conceptual foundation

for action patterns are so-called behavioral profiles, which capture constraints on the level of activity pairs. A behavioral profile builds on trace semantics for a process model, namely the weak order relation [20]. It contains all pairs (x, y) if there is a trace in which x occurs before y. For a process model p, we write $x \succ_p y$. The behavioral profile then defines a partition over the Cartesian product of activities, such that a pair (x, y) is in one of the following relations:

- strict order relation $\rightsquigarrow_p$, if $x \succ_p y$ and $y \not\succ_p x$;
- exclusiveness relation $+_p$, if $x \not\succ_p y$ and $y \not\succ_p x$;
- interleaving order relation $||_p$, if $x \succ_p y$ and $y \succ_p x$.

Based on this behavioral profile, we define the behavioral constraints of a service candidate. Therefore, we consider each behavioral action pattern as a rule, and compute its minimum support and confidence value [19]. By only considering object-sensitive patterns, i.e., patterns focusing on actions of the same business object, we obtain the basis for detailing the lifecycle of a composite service candidate. We then use the synthesis technique defined in [21], to derive a process model based on the behavioral relationships. The strict order relations define the skeleton of a corresponding process model. Activities that are not in an order relation are organized in nested XOR- or AND-blocks depending on whether they are exclusive or interleaving. Note that this concept can also be applied to action-based composite services. To this end, only action-sensitive patterns must be considered.

4 Evaluation

To demonstrate the applicability of our service identification approach, we conduct an evaluation with real-world data. In particular, we designed a test collection that contains the activity labels of the SAP Reference Model. The SAP Reference Model is a collection of Event-Driven Process Chains (EPCs) and captures the business processes supported by the SAP R/3 system in its version from the year 2000 [22, pp. 145-164]. It is organized in 29 functional branches as for instance sales and accounting and contains 604 process models with in total 2433 activity labels. In the following paragraphs we present the results for each introduced identification technique.

4.1 Atomic Service Identification Results

For obtaining atomic service candidates, we computed the metric $F_{Activity}$ for each activity in the process model collection. As a result, we identified 464 activities with a frequency of at least two. Twelve of these activities had a frequency of equal or greater 10. Table 1 gives an overview of the top 5 ranked results.

The results show that it is still necessary to evaluate whether an identified candidate is suitable for being established as a service. In addition, we must decide whether a candidate can be established as a business or as a software service. For instance, *Process Goods Issue*, *Billing* and *Planning* are more likely to

represent business services, while *Calculate Overhead* and *Difference Processing* could be automatable activities and are hence candidates for software services.

Table 1. Results for Atomic Service Identification

Rank	Activity	$\mathbf{F_{Activity}}$
1	Process Goods Issue	20
2	Calculate Overhead	17
3	Billing	13
4	Planning	13
5	Difference Processing	13

4.2 Composite Service Identification Results

Based on the consideration of the metric F_{Object}, we identified 378 business object groups with a minimum occurrence of two. In 27 of these groups, the business object was used ten times or more. Table 2 shows the top 5 ranked business objects groups. In addition to the rank and the value of F_{Object}, it also provides the most frequent actions that are applied on these business objects in the analyzed models.

Table 2. Results for Composite Service Identification

Rank	Business Object	Example Actions	$\mathbf{F_{Object}}$
1	Order	Execute, Settle, Archive, Release, Print	48
2	Time Sheet	Report, Permit, Process, Approve, Create	23
3	Invoice	Release, Verify, Process, Receive, Reverse	23
4	Budget	Release, Plan, Update, Allocate, Return	23
5	Posting	Perform, Release, Direct	19

4.3 Inheritance Hierarchy Service Identification Results

To extract a business object hierarchy, we derived the different parts for each business object and computed their frequencies. In this way, for instance, the business object *Service Product Order* was first reduced to *Product Order* and then to *Order*. Whenever a sub term was identified twice, a new node in the business object hierarchy was created. Taking the given example, only a new node for *Order* is introduced as the term *Product Order* only appeared once among all activities of the model collection. By computing the metric $F_{Hierarchy}$ for each main word, we obtained a ranked list of business object hierarchies.

In total, we identified 362 business object hierarchies. In 70 hierarchies the root term was used 10 times or more. Table 3 shows the top 5 ranked business object hierarchies including the main word and three frequent example nodes.

Table 3. Results for Composite Service Identification

Rank	Main Word	Example Nodes	$F_{Hierarchy}$
1	Order	Business Order, Sales Order, Service Order	112
2	Data	Plan Data, Transaction Data	51
3	Cost	Plan Cost, Process Cost, Shipment Cost	46
4	Request	Payment Request, Recruitment Request	30
5	Document	Billing Doc., Customer Doc., Customer Doc.	29

4.4 Action-based Composite Service Identification Results

Based on the consideration of the metric F_{Action}, we identified 221 action groups with a minimum occurrence of two. In 54 of these groups, the action was used ten times or more. Table 4 shows the top 5 ranked action groups. In addition to the rank and the value of F_{Action}, it also provides the most frequent business objects that occurred with these actions.

Table 4. Results for Action-based Composite Service Identification

Rank	Action	Example Objects	F_{Action}
1	Process	Goods Issue, Difference, Goods Receipt	397
2	Plan	Cost, Material, Budget	147
3	Transfer	Data, Objects, Results	54
4	Calculate	Overhead, Interest, Account Balance	52
5	Settle	Order, Project, Consignment	51

4.5 Computation of Internal Service Structure

To determine the internal structure of a composite service, we computed the behavioral profile for the comprised activities. Table 5 shows the behavioral profile for the composite service *order*. This profile illustrates that there exists a well-defined order in which the activities must be executed. Apart from the activity *print*, which can be performed at any time after the activity *process*, the order is strict. The synthesis of this profile yields the process model depicted in Figure 2.

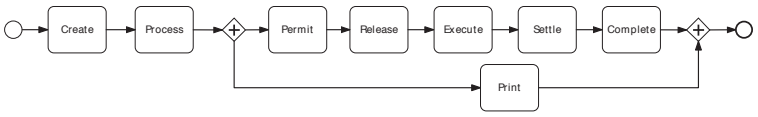

Fig. 2. The process model for the composite service *order*

Table 5. The behavioural profile for the composite service *order*

	create	*process*	*print*	*release*	*execute*	*permit*	*settle*	*complete*
create		$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
process			$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
print				$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
release					$\rightsquigarrow$	$\rightsquigarrow^{-1}$	$\rightsquigarrow$	$\rightsquigarrow$
execute						$\rightsquigarrow^{-1}$	$\rightsquigarrow$	$\rightsquigarrow$
permit							$\rightsquigarrow$	$\rightsquigarrow$
settle								$\rightsquigarrow$

5 Related Work

The work presented in this paper relates to approaches for service identification and the application of natural language processing for process models.

Service identification has been considered for business services and software services [2]. The identification of business services is mainly discussed by papers from practice [23,24]. Those typically build on the analysis of business entities and components. By contrast, the derivation of software services is widely discussed in research. Some authors propose a bottom-up approach by analyzing existing legacy systems and building on their functionality [8,9,10]. Others suggest a top-down strategy [25,26]. However, most approaches proposes to strike a balance between the two extremes. As a result, plenty of software service identification approaches include the analysis and evaluation of business process models [6,7,13,14].

Techniques for natural language processing have been applied on process models in various contexts. One important application scenario is the assurance of linguistic quality aspects in process models. With this intention NLP tools were employed for identifying semantic errors in activity labels [27] and for constructing a glossary from process model collections [28]. NLP tools were also used for refactoring whole process model collections to ensure the understandability of the comprised activity labels [15]. As the latter requires the decomposition of the activity label into its components, this technique also constituted the basis for the approach presented in this paper. Other prominent applications of NLP techniques are the identification of similarities between process models [29,30] and the derivation of process models from natural language texts [31,32,33].

6 Conclusion

In this paper, we presented an approach for the automatic identification of service candidates from process models. We built on analysis techniques for process models and proposed four different techniques for deriving service candidates. We tested our approach on a process model collection from practice including over 600 EPCs with in total 2433 activities. The evaluation illustrates the capability of

our algorithm to provide useful service candidates. Our technique does not only enable companies to take an extensive number of process models into account, but also to efficiently analyze them. Considering the results, it is important to emphasize that our technique cannot completely automate the service derivation procedure because the final decision making remains a human task.

In future research, we plan to extend our technique by incorporating lexical relationships such as synonymy and homonymy. Further, we aim to apply our technique in the context of an industrial case study. In this way, we aim at determining the applicability and the significance of each of the identification strategies. In response to the findings, the proposed approach could then be adapted to the specific needs from practice.

References

1. O'Sullivan, J., Edmond, D., ter Hofstede, A.H.M.: What's in a service? Distributed and Parallel Databases **12**(2/3) (2002) 117–133
2. Kohlborn, T., Korthaus, A., Chan, T., Rosemann, M.: Identification and analysis of business and software services - a consolidated approach. IEEE T. Services Computing **2**(1) (2009) 50–64
3. Adamopoulos, D., Pavlou, G., Papandreou, C.: Advanced service creation using distributed object technology. IEEE Comm. Magazine **40**(3) (2002) 146–154
4. Chang, S., Kim, S.: A systematic approach to service-oriented analysis and design. In: Product-Focused Software Process Improvement. LNCS 4589 (2007) 374–388
5. Bianchini, D., Cappiello, C., Antonellis, V., Pernici, B.: P2s: A methodology to enable inter-organizational process design through web services. CAiSE '09, Berlin, Heidelberg, Springer-Verlag (2009) 334–348
6. Kleinert, T., Balzert, S., Fettke, P., Loos, P.: Systematic identification of service-blueprints for service-processes - a method and exemplary application. In Rosa, M., Soffer, P., eds.: BPM Workshops. Volume 132 of LNBIP. (2013) 598–610
7. Azevedo, L., Santoro, F., Baião, F., Souza, J., Revoredo, K., Pereira, V., Herlain, I.: A method for service identification from business process models in a soa approach. In: Enterprise, Business-Process and Information Systems Modeling. Volume 29 of LNBIP. Springer Berlin Heidelberg (2009) 99–112
8. Aversano, L., Cerulo, L., Palumbo, C.: Mining candidate web services from legacy code. In: 10th International Symposium on Web Site Evolution. (2008) 37–40
9. Chen, F., Zhang, Z., Li, J., Kang, J., Yang, H.: Service identification via ontology mapping. IEEE 36th Annual Computer Software and Applications Conference **1** (2009) 486–491
10. Zhang, Z., Liu, R., Yang, H.: Service identification and packaging in service oriented reengineering. In: Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering. (2005) 241–249
11. Leopold, H., Mendling, J.: Automatic derivation of service candidates from business process model repositories. In: Proceedings of the 15th Int. Conf. BIS. Volume 117 of Lecture Notes in Business Information Processing, Springer (2012) 84–95
12. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal **12**(2) (2006) 249–254
13. Klose, K., Knackstedt, R., Beverungen, D. In: Identification of Services - A Stakeholder-Based Approach to SOA Development and its Application in the Area of Production Planning. University of St. Gallen (2007)

14. Erradi, A., Kulkarni, N., Maheshwari, P.: Service design process for reusable services: Financial services case study. In: Proceedings of the 5th international conference on Service-Oriented Computing. ICSOC '07, Berlin, Heidelberg, Springer-Verlag (2007) 606–617
15. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems **37**(5) (2012) 443–459
16. Leopold, H., Eid-Sabbagh, R.H., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems **56** (2013) 310–325
17. Inaganti, S., Behara, G.K.: Service identification: Bpm and soa handshake. BPTrends (2007)
18. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. **2** (July 2006) 412–442
19. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process model repositories. Computers in Industry **63** (2012)
20. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. **37**(3) (2011) 410–429
21. Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on synthesis from consistent behavioural profiles. International Journal of Cooperative Information Systems **21** (2012)
22. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley (1998)
23. IBM: Component business models (2005)
24. SAP: Enterprise service design guide (2005)
25. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA (2005)
26. Flaxer, D., Nigam, A.: Realizing business components, business operations and business services. In: Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference. CEC-EAST '04, Washington, DC, USA, IEEE Computer Society (2004) 328–332
27. Gruhn, V., Laue, R.: Detecting Common Errors in Event-Driven Process Chains by Label Analysis. Enterprise Modelling and Information Systems Architectures **6**(1) (2011) 3–15
28. Peters, N., Weidlich, M.: Automatic Generation of Glossaries for Process Modelling Support. Enterprise Modelling and Information Systems Architectures **6**(1) (2011) 30–46
29. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. **36** (April 2011) 498–516
30. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: APCCM 2007. Volume 67., Ballarat, Victoria, Australia, Australian Computer Science Communications (2007) 71–80
31. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: CAiSE 2011. Volume 6741 of LNCS., Springer (2011) 482–496
32. de AR Gonçalves, J.C. and Santoro, F.M. and Baião, F.A.: Business Process Mining from Group Stories. In: CSCWD 2009, IEEE Computer Society (2009) 161–166
33. Sinha, A., Paradkar, A.: Use Cases to Process Specifications in Business Process Modeling Notation. In: 2010 IEEE International Conference on Web Services, IEEE (2010) 473–480

Being Digital: The Power of the Bits. Science, University and Industry Perspective

Jesús Bermejo Muñoz[1], Carlos Fernández Sánchez[2], and Jordi Guijarro Olivares[3]

[1]SGi Users' Forum Spain
http://spain.osgiusers.org/
jesus@bermejo.link

[2]Supercomputing Centre of Galicia
https://cesga.es/en/cesga
carlosf@cesga.es

[3]Consorci de Serveis Universitaris de Catalunya
http://www.csuc.cat/en
jordi.guijarro@csuc.cat

Abstract. This book, made of "unwieldy atoms", will probably be replaced by a digital copy by the time anyone reads it, stated Nicholas Negroponte when presented “Being Digital” in 1995. Nevertheless, the process of digitization during the last two decades has probably moved far beyond what he could have imagined at that time.

Currently, there is a wide consensus on the fact that we are living a major transition, and that information technology (IT) powered services will have a relevant impact in terms of employment, business and economy. Computing, data analytics and, more in general, IT powered services will play an everyday more active role in our lives. The current speed of the technology is leading to foresee that business in the future will be carried out differently than today.

However, the potential of IT technology is still far from being efficiently applied addressing key social challenges such as sustainability, education, health and many others currently challenging future generations. Existing barriers among science, university and industry innovation approaches, combined with the current speed of the technologies in the field, are limiting the development of efficient innovation ecosystems.

The speech, jointly prepared by representatives from science, university and industry, will analyse the current digitalisation process in these sectors, current challenges in science-university-industry innovation cycle and collaborative efforts trying to address the existing gaps. It also considers a slot for discussions/debate within software and services research community.

A Mobile End-User Tool for Service Compositions *

Ignacio Mansanet, Victoria Torres, Pedro Valderas, and Vicente Pelechano

Universitat Politècnica de València, Spain
Pros Research Center
{vtorres,imansanet,pvalderas,pele}@pros.upv.es

Abstract. With the advent of Web 2.0 and the massive adoption of smartphones, end-users are more keen to actively participate in the creation of content for the Internet. In addition, the big amount of data that the Internet of Things can bring to it, establishes an ideal framework to allow end-users not just consuming data but also creating new value-added services. To achieve this, in this work we propose the development of an end-user tool for smartphones capable of composing services that are available in the Internet.

1 Introduction

Since 2004 where the term Web 2.0 was popularized, the role played by end-users in the development of the web has dramatically changed. Since then, the availability of easy-to-use tools such as the ones developed within social networks, wikis, CMSs, etc. have promoted the interaction and collaboration of end-users to generate new content.

With the advent of the Internet of Things (IoT), a big amount of objects can be virtually represented in the Internet (by 2020 Gartner estimates nearly 26 billion of devices on the IoT[2]). These objects will offer data about their state; e.g., temperature, lightning, and sound pressure level will be provided by thermoters, lightning detectors, and sound meter levels respectively. These data are usually provided by web services which collect them by querying the corresponding source (i.e., the specific device or the database that stores them). Even though these services can be used individually, their composed usage has the potential to create new value-added services. However, the creation of such compositions requires users with programming background since existing composition environments (such as Intalio, Activiti, Signavio or Bonita BPM) and service composition languages or notations (such as Petri nets, EPC, YAWL, BPMN or UML Activity Diagrams) are not yet targeted to ordinary users without such skills.

* This work has been developed with the support of the Spanish Ministry of Economy and Competitivity IPT-2012-0839-430000 and the Valencian Government ACOMP/2014/186 and financed jointly with ERDF

[2] http://www.gartner.com/newsroom/id/2636073

Moreover, existing environments to create such compositions have been traditionally designed to be used from desktops or laptops. However, the type of devices that end-users may use are not longer of this type but mobile devices such as tablets or smartphones. According to Gartner[3], in the second quarter of 2013 worldwide smartphone sales to end-users reached a total of 225 million units. Compared to the second quarter of 2012, this figure represents an increase of 46.5 percent. Furthermore, worldwide smartphones sales have overtaken for the first time feature phone sales, constituting the 51.8 percent of mobile phone sales. All these figures confirm the smartphone mass adoption by end users. According to this evidence, make sense to think that end-users will use such devices not just for consuming data and services but also to create new ones either from scratch or from content that is already made available in the Internet.

Within this context the current work presents an end-user tool for the construction of service compositions by using mobile devices. To this end, the development of this tool has been inspired by change patterns and techniques for end-user development (EUD). This work is being developed within the context of SITAC (Social Internet of Things, Apps by and for the Crowd), an ITEA 2 project aimed at the convergence of the IoT and Social Networks to enable a better interaction among humans and devices.

The remainder of the paper is structured as follows. Sect. 2 presents a scenario that illustrates the practical application of the presented end-user tool. Then, Sect. 3 introduces the foundations of the tool, which are change patterns and end-user development. Sect. 4 explains in detail the design and implementation of the end-user tool. Sect. 5 presents some related work. Finally, Sect. 6 concludes the paper and provides insights of directions for future work.

2 Running Example

In order to exemplify the proposal, we present an example that will be used through the paper. The scenario describes a day in John's life. John is a 22 years old university student who is actually enrolled in his 4th year of studies. The university where John is studying offers some IoT services which provide the university community with information about university facilities (i.e., library availability, parking places availability, events celebrated in a specific date, etc.). During the exams period, John likes to go to the university library to study and meet his classmates. To do so, John is using the university facilities as follows. Before leaving home, John searches for libraries where there are still available seats and for bike parks closeby with free space for his bike. Based on the results of these searches John selects a library and a bike park and book places for him and his bike. In addition, while performing the booking, John notifies his classmates about the library where his is going to stay publishing this information via his Facebook account. Fig. 1 shows the BPMN process that describes all the tasks performed by John. Since John does these tasks every

[3] http://www.gartner.com/document/2573119

morning during this period, he wants to define a composition service linking all these services properly.

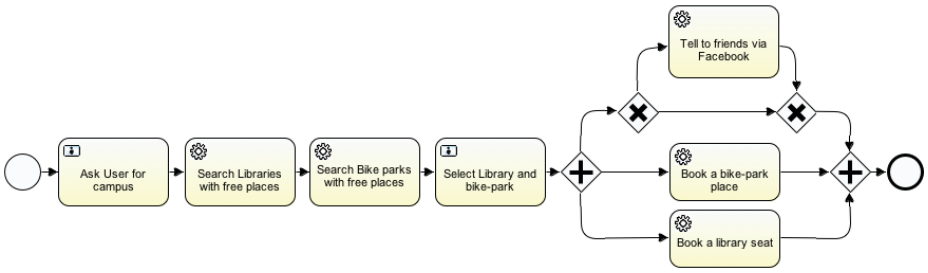

Fig. 1. Service composition for the running example specified in BPMN

3 Foundations

This section introduces the two research fields in which this work relies on. On the one hand, it presents the different techniques that have been developed within the EUD field. On the other hand, it presents change patterns and the advantages that these introduce when building process models, which are seen as descriptions of service composition at a higher level of abstraction.

3.1 End-user Development

The objective of EUD is to develop tools that can be successfully used for different types of users, i.e., novices and experts. While novices should find these tools easy to use, not be intimidating and giving them the enough confidence to succeed, experts should be able to work on sophisticated and completed solutions.

Many techniques have been developed in EUD for the development of applications. These techniques include programming-by-example, visual or textual languages. Programming-by-example is a programming technique where the user creates an application by performing the steps as an example. However, it is not easy to create the final application using this technique in isolation, since this do not allow end-users reviewing, testing, and debugging applications. To alleviate this problem, the programming-by-example technique is used in combination with visual or textual languages. For example, visual programming allows simplifying the development of applications since it provides users with environments that provide constructs in a higher level of abstraction, hiding the most tedious technical part. In this line of research, dataflow visual language is one paradigm that has been often applied. An example of this technique is Yahoo! Pipes (http://pipes.yahoo.com/) that provides an environment capable of aggregating, manipulating and mashing up content from the Web.

On top of these techniques there are recommendations or design guidelines aimed at helping in the development of such tools [1]. These recommendations deal with syntactic, semantic, and pragmatic aspects that need to be taken into consideration to develop effective end-user tools. Well-known design suggestions and patterns [2, 3, 4, 5] in the area of EUD include (1) the usage of the catalogue pattern, which offer end-users a set of predefined elements to face the selection barriers [3], (2) tabs which improve the learning and memorization of the steps to be performed and keeps end-users from missing important things [4], (3) textual descriptions of the results in order to facilitate its understanding, or (4) designing all the interfaces following a similar structure to facilitate user interpretation of the information.

In the EUD community, it is accepted that the use of wizards can save end-users from a potentially frustrating experience [4]. For this reason, for the development of our end-user tool we rely on wizards following the design suggestions and patterns presented above. Wizards will guide users through a particular process with the objective of creating service compositions.

3.2 Change Patterns

Change patterns are high level abstractions aimed at achieving flexible and easy adaptations of business process [6]. These abstractions are defined in terms of high level change operations (e.g., the creation of a parallel branch) which are based on the execution of a set of change primitives (e.g., add/delete activity). As opposed to change primitives, change pattern implementations typically guarantee model correctness after each transformation [7] by associating pre-/post-conditions with high-level change operations. In process modeling environments supporting the correctness–by–construction principle (e.g., [8]), usually process modelers only have those change patterns available for use that allow transforming a sound process model into another sound one. For this purpose, structural restrictions on process models (e.g., block structuredness) are imposed. In addition, a correct usage of change patterns allows speeding up the creation of the composition (while the construction of Fig. 1 requires 31 steps with change primitives, just 12 steps are needed when using change patterns).

In this work we take as basis from [6] a small subset of change patterns to allows users to create all main control-flow constructs (i.e., sequences, parallel branches, conditional branches, and loops). This set includes: AP1 (Insert Process Fragment) with its two variants, i.e., Serial and Parallel Insert, AP2 (Delete Process Fragment), AP8 (Embed Fragment in Loop), and AP10 (Embed Process Fragment in Conditional Branch). For illustration purposes, Fig. 2 shows the changes introduced by the application of pattern AP8 into S to obtain S' where activity B is embedded in a loop.

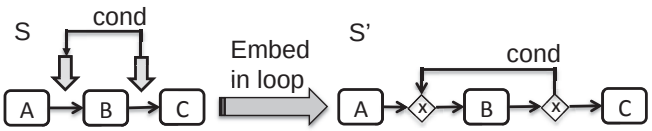

Fig. 2. Pattern AP8: Embed Process Fragment in Loop

4 End-user Tool for Service Composition

This section presents the Domain Specific Language (DSL) that has been designed to implement the end-user tool for service composition. While its abstract syntax has been defined based on the change patterns introduced in Sect. 3.2, its concrete syntax has been defined based on the wizard concept.

4.1 Domain Specific Language: Abstract Syntax

Fig. 3 shows the metamodel that defines the DSL that allow specifying service compositions in a high level of abstraction. Inspired by the set of patterns presented in Sect. 3.2, the metamodel is made up of two main concepts which are *activities* and *fragments*. This facilitates their definition by end-users since they only have to understand these two concepts. *ServiceElements* are defined through the composite pattern, which facilitate the creation of user interfaces that help end-users to focus on the definition of either the service main sequence or the content of a fragment, individually. The *previousElement* relationship between *ServiceElements* allows establishing the sequence order between *activities* and *fragments*. We thought that it would be easier for end-users without notions of process modelling to create nodes associated to a previous one than explicitly create flow elements. In addition, this sequence order implicitly defines the start and end of the composition, being the first and last *activity* or *fragment* the start and the end of the composition respectivelly. Finally, *services* and *parameters* are complemented with a description attribute, whose main goal is to provide end-users with a textual description that helps them to understand the semantics of a service or a parameter.

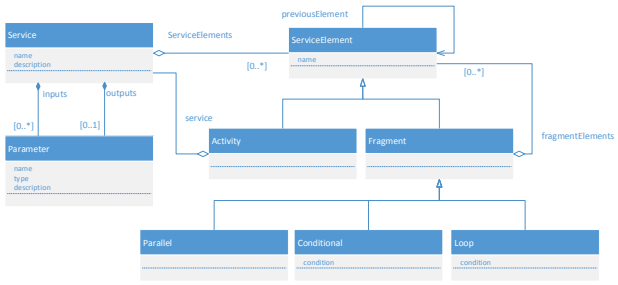

Fig. 3. Domain Specific Language

In order to better understand the concepts of this metamodel, Fig. 4 identifies them in the process used as a running example. This process is composed of a sequence of four activities followed by a parallel fragment. In turn this fragment is made up of a conditional fragment and two activities being executed in parallel to it.

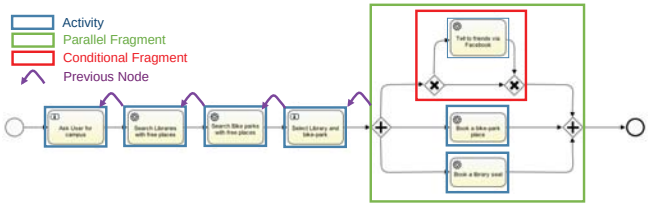

Fig. 4. DSL concepts applied to the running example

4.2 Domain Specific Language: Concrete Syntax

In this section, we introduce the concrete syntax of the end-user tool, which is based on the use of wizards. The primitives used in the wizards are in accordance with the abstract syntax presented above. The implementation of this wizard has been performed by applying styles and designs for mobile devices. To achieve this, we have used the jQuery Mobile framework[4]. In particular, wizards have been defined considering the following well-known design suggestions and patterns (see Fig.5).

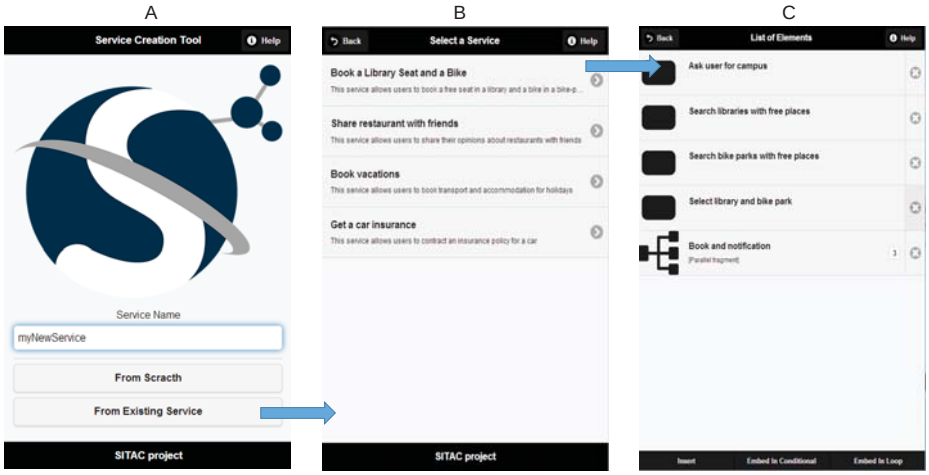

Fig. 5. Wizard to create a service composition from an existing service

[4] http://jquerymobile.com

Catalog pattern. We use the catalog pattern to offer a set of predefined elements, services in our case, to end-users. A detailed description of these elements is presented in Fig. 6A. From left to right, each service element (activity or fragment) is described by three graphical components: (1) an icon, which allows identifying the type of element being represented (activity or fragment, and in case of a fragment, whether this is parallel, conditional, or loop); (2) the name of the element; and (3) the number of elements included in a fragment (just for fragments). Note that the order in which tasks and fragments appear in this interface indicates the sequence order between them.

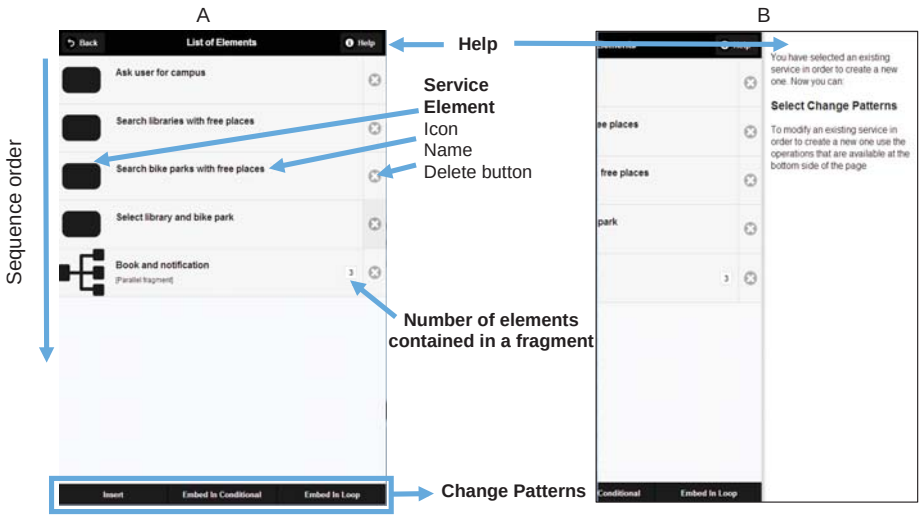

Fig. 6. Details of a service composition

Tabs. We guide users by using tabs that indicate what they have already done and what they can do along the wizard. As Fig. 6B shows, each user interface has a Help button that allow users to access a tab that explains the actions that end-users have already done and the actions that they can do from this point.

Change patterns. To create service compositions, end-users can make use of any of the available change patterns that are implemented in the tool. These patterns can be access from the buttons that are provided in the *list of Elements* user interface (cf. Fig. 6A) at the bottom side of the page, i.e., Insert (AP1–Insert Process Fragment), Embed in Conditional (AP10–Embed Process Fragment in Conditional Branch), and Embed in Loop (AP8–Embed Fragment in Loop)). Differently from these patterns, the Delete pattern (AP2–Delete Process Fragment) is directly used from the corresponding element by clicking on the x button placed at its right-hand side (cf. Fig. 6A). If end-users select to embed an element in a conditional branch or in a loop pages A and B in Fig.7 are accessed, respectively. In both cases, end-users must indicate a name and a condition. In

the case of a conditional branch, the elements that must be executed when the condition is satisfied must be selected. In the case of a loop fragment, end-users must select the first and last element in the loop in order to indicate that after the last element, if the condition is satisfied, the flow of the service must go to the first element. Note that both pages show a textual description of the result at the bottom side.

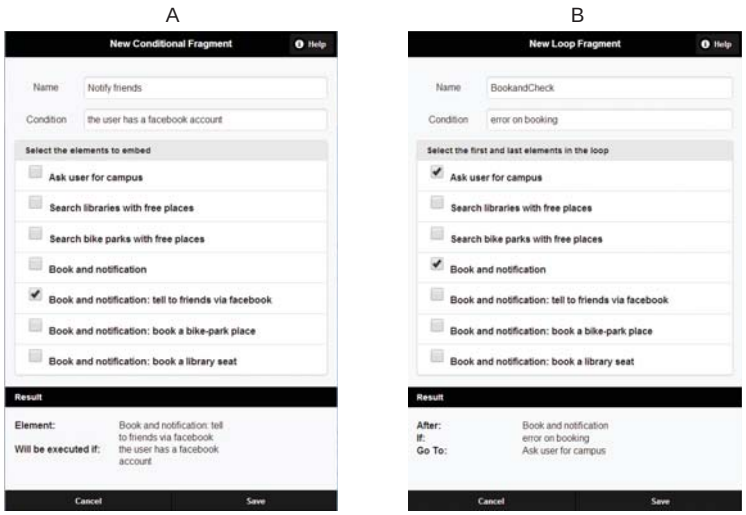

Fig. 7. Embeding elements in a conditional branch or loop

When end-users select to insert a service element the page in Fig. 8A is shown. From this page, end-users can select to insert an activity or a parallel fragment. If they select the first option, the page shown in Fig. 8B is accessed. Then, end-users can create and activity by indicating a name, the previous element, and a predefined service. The predefined service is selected from the catalogue of existing ones that is supported by the tool. Note that a textual description of the result of this insert action is shown at the bottom side of the page. To finish this step, the inputs of the predefined service must be configured. To do so, end-users can access the page in Fig. 8C, which allows them to indicate if inputs must be introduced by users or must be defined from the output of an already defined service element (such as in Fig. 8C). In this last case, end-users can select the outputs of a service element that will be used as inputs.

If end-users select to insert a parallel fragment page in Fig. 9A is shown. From this page, a parallel fragment can be created by indicating a name, a previous element, and the elements that must be executed in a parallel way. Note how a textual description of the results is shown at the bottom side of the page. When a parallel fragment is created it is inserted in a sequence of the service as it is shown in Fig. 9A. From this sequence, the fragment can be selected in order to access the page in Fig. 9C, which show the elements that it includes. From this

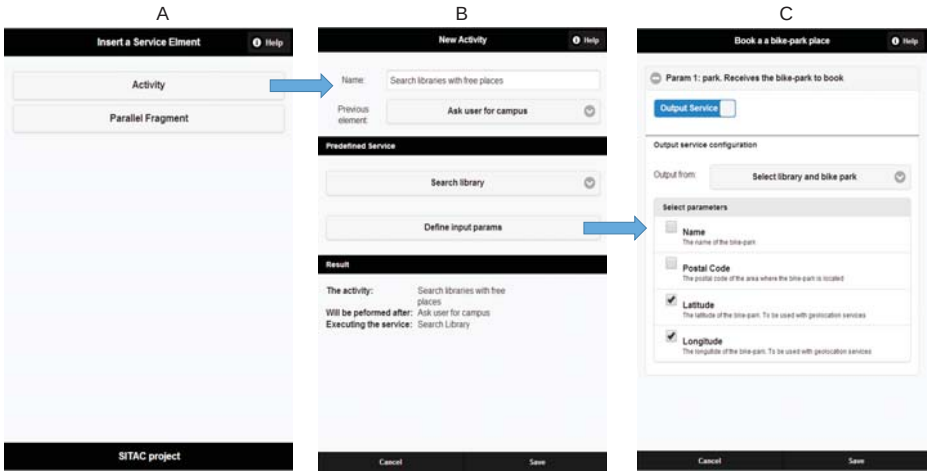

Fig. 8. Inserting an activity

page, the change patterns (Insert, Embed in Loop, Embed in Conditional) can be applied to the content of the fragment.

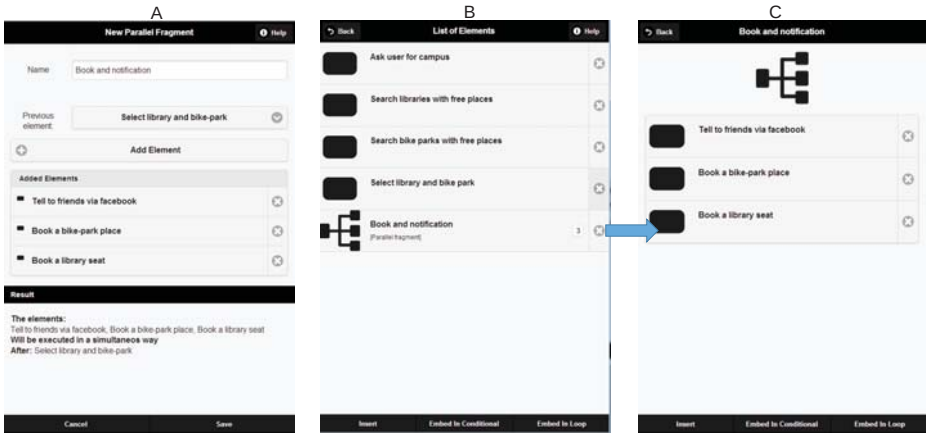

Fig. 9. Inserting a parallel fragment

Textual descriptions. Actions done by end-users are complemented with textual descriptions of the results in order to facilitate its understanding.

Similar structures. All the interfaces (as Fig. 5-9 show) share a similar structure to facilitate user interpretation of the information.

5 Related Work

The mass adoption of mobile devices by end-users together with the amount of services available in Internet has motivated the development of research of Mobile mashups, a field aimed at the development of tools that allow end-users the aggregation, manipulation, and mashuping content from around the web using such devices.

Similarly to our end-user tool, existing research in this field proposes the development of tools based on well-known EUD techniques. By using different Visual languages, these works propose authoring tools aimed at the construction of applications based on services offered by the device at hand or services coming from external services. For example, [9] proposes SCK (Service Creation Kit), an authoring prototype tool for the development, deployment and execution of microservices (small applications that allow users to obtain and provide information from fellow users). Antoher example aimed at the construction of mobile applications is [10]. Here authors propose the generation of mobile applications by specifying the sequence of actions needed to model the required application. Finally, in [11] authors propose an EUD framework based on model-driven development techniques. This framework is aimed at the design and automatic generation of mobile mashups.

We argue that the main difference between our work and all these proposals is that the design of our end-user tool relies on change patterns. As we explained in Sect 3.2, the two major advantages of using change patterns are that ensure correctness-by-construction and that they speed up the construction process. In addition, these patterns allow us to provide end-users a more powerful tool that allows creating service compositions using not just sequences or paralallel branches but also embedding fragments into conditional and loop structures.

6 Summary and Future Work

In this work we have presented an EUD tool aimed at the composition of services by means of a smpartphone. The main objective of this tool is to provide end-users with all the necessary functionality to create service compositions based on the services provided by the SITAC platform. This functionality has been designed based on a specific set of change patterns and implemented using recommendations and guidelines from the EUD field. However, such service compositions are not yet executable compositions. In fact, end-users did not have to deal with technological details (e.g. BPMN, REST, XML) that are needed to execute such compositions. To achieve this, the end-user tool will be complemented with a set of components (e.g., data mappings and transformations, invoking REST services, etc.) that are needed to execute them in a process engine. Moreover, the idea is to automate as much as possible the tasks performed by these complementary components so they remain transparent to the end-user.

As future work we plan to assess, by means of the Technology Acceptance Model (TAM), end-users' perceived usefulness and perceived ease of use. In addi-

tion, these subjective measures will be contrasted with users' actual performance with think-aloud sessions which will allow us to analyze the problem-solving processes followed by end-users during the usage of the tool.

References

1. Repenning, A., Ioannidou, A.: What makes end-user development tick? 13 design guidelines. End User Development **9** (2006) 1–41
2. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37** (2005) 316–344
3. Ko, A.J., Myers, B.A., Aung, H.H.: Six learning barriers in end-user programming systems. In: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing. VLHCC '04, Washington, DC, USA, IEEE Computer Society (2004) 199–206
4. van Welie, M., Traelig;tteberg, H., Trtteberg, H.: Interaction patterns in user interfaces. In: Proc. Seventh Pattern Languages of Programs Conference: PLoP 2000. (2000) 13–16
5. Mick, C.P., Roger, T., Frederick, C.G., Scott: What They See Is What We Get: Response Options for Web Surveys. Social Science Computer Review **22** (2004) 111–127
6. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering **66** (2008) 438–466
7. Casati, F.: Models, Semantics, and Formal Methods for the design of Workflows and their Exceptions. PhD thesis, Milano (1998)
8. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comp Scie - R&D **23** (2009) 81–97
9. Danado, J., Davies, M., Ricca, P., Fensel, A.: An authoring tool for user generated mobile services. In Berre, A., Gmez-Prez, A., Tutschku, K., Fensel, D., eds.: Future Internet - FIS 2010. Volume 6369 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 118–127
10. Cuccurullo, S., Francese, R., Risi, M., Tortora, G.: Microapps development on mobile phones. In Costabile, M., Dittrich, Y., Fischer, G., Piccinno, A., eds.: End-User Development. Volume 6654 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 289–294
11. Cappiello, C., Matera, M., Picozzi, M.: End-user development of mobile mashups. In Marcus, A., ed.: Design, User Experience, and Usability. Web, Mobile, and Product Design. Volume 8015 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 641–650

Alineación de modelos de negocio y software: un método orientado a servicios centrado en la arquitectura

Marcos López-Sanz, Valeria de Castro, Esperanza Marcos
Grupo de Investigación Kybele, Universidad Rey Juan Carlos
Tulipán S/N, 28933, Móstoles, Madrid
{marcos.lopez, valeria.decastro, esperanza.marcos}@urjc.es

Resumen. La alineación de negocios con soluciones tecnológicas orientadas a servicios se ha mostrado como un aspecto de vital importancia en la empresa moderna. En este sentido, la provisión de métodos para solventar el salto de negocio a tecnología se hace totalmente necesaria. Este artículo presenta una propuesta que pretende sistematizar ese salto mediante la definición de un método de desarrollo centrado en la arquitectura. La utilización de diferentes modelos arquitectónicos a diferentes niveles de abstracción junto con la definición de transformaciones entre modelos permite establecer una traza entre elementos de nivel de negocio y los elementos software que se deriven de ellos como soporte tecnológico. Los beneficios clave de nuestra propuesta son, por un lado, la provisión del método en sí para la alineación negocio-tecnología y, por otro lado, la definición de un nuevo modelo para representar la estructura de un negocio. Esta propuesta ha sido refinada utilizando el caso de un sistema de información para la gestión de percentiles pediátricos.

Palabras clave. Alineación negocio-tecnología, Procesos de desarrollo centrados en la arquitectura, Orientación a servicios, Desarrollo dirigido por modelos.

1 Introducción

Muchas empresas están evolucionando hacia un esquema económico basado en la prestación de servicios a través de la definición clara del conjunto de capacidades que ofrecen [7]. Este esquema ha progresado de tal forma que los servicios se presentan con frecuencia como parte integrante fundamental de la realización actual de negocios [22]. En este escenario orientado a los servicios, la tecnología se ha convertido en un aliado necesario y rentable. Así, durante la última década, la orientación a servicios se ha extendido hasta convertirse en uno de los paradigmas de computación más importantes en el seno de las empresas modernas [11]. El concepto de servicio ha servido para alinear los negocios con sistemas software de soporte adecuados [4]. En ambos ámbitos (empresarial y software), la identificación de los participantes y la naturaleza de sus relaciones es un aspecto clave. A este respecto, la *arquitectura* se puede utilizar como artefacto para progresar desde el dominio del negocio al del software, utilizando la orientación a servicios como base paradigmática [14].

El uso de modelos arquitectónicos para representar las arquitecturas de negocio y de software es una cuestión importante que puede facilitar la alineación de los negocios con un software de soporte apropiado [10]. Esta circunstancia es incluso más

eficaz en entornos orientados a servicios donde el concepto de servicio se utiliza para representar entidades en ambos dominios [2].

En este trabajo presentamos un método centrado en la arquitectura para la alineación de modelos de negocio y software orientado a servicios. La idea fundamental es la utilización de modelos arquitectónicos como artefacto central de un proceso de desarrollo de software. Además, nuestra propuesta de alineación utiliza un enfoque dirigido por modelos [6], es decir, contiene la definición de modelos, y la especificación de reglas de transformación entre ellos. El uso de técnicas basadas en modelos, y más específicamente el uso de transformaciones de modelos permite reducir la brecha entre servicios de negocio y los servicios tecnológicos [3].

En este artículo vamos a ilustrar la viabilidad de la propuesta mediante su aplicación a un caso de estudio del mundo real. El caso de estudio se basa en el desarrollo de un sistema de información para la gestión de percentiles de crecimiento que facilite a los médicos pediatras el seguimiento de sus pacientes.

El resto de este trabajo se organiza de la siguiente manera: la sección 2 presenta una visión general del método y de los modelos incluidos. En la sección 3 se presenta el proceso de alineación propuesto aplicado a un caso de estudio. La sección 4 presenta un análisis de algunos de los trabajos relacionados y la sección 5 concluye el artículo y muestra algunos trabajos futuros derivados de la investigación realizada.

2 Descripción general del método y modelos incluidos

Como se muestra en el lado izquierdo de la Fig. 1, nuestra propuesta sitúa la arquitectura como aspecto central del proceso de desarrollo. A nivel de negocio se define el modelo de la *arquitectura de la estructura de negocio*. La arquitectura software se define a un nivel de abstracción inferior, donde entra en relación con otros aspectos de software como la interfaz o el comportamiento. El lado derecho de la Fig. 1 muestra los modelos concretos involucrados en la propuesta. Esta separación en niveles de abstracción se realiza de acuerdo con MDA [15] e incluye la definición de un conjunto de reglas de transformación de modelos.

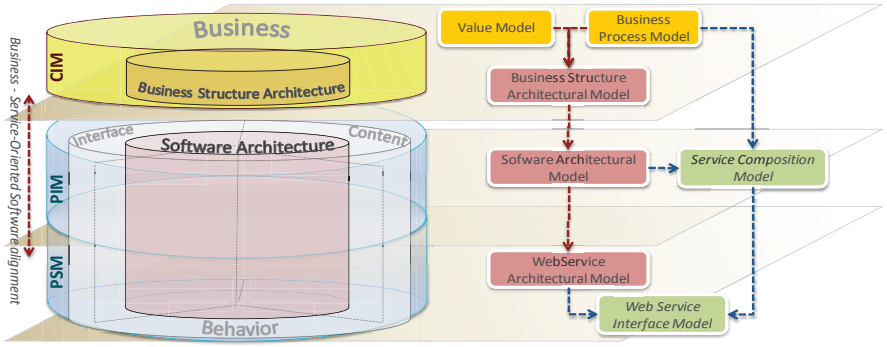

Fig. 1. Vista general del método centrado en la arquitectura.

Presentamos a continuación los modelos propuestos para representar, por separado, el contexto de negocio, el modelado de la arquitectura, y el aspecto de comportamiento:

- **Modelado del negocio**. La información que se refleja en este nivel se refiere a las actividades comerciales y los intercambios de valor que se producen como parte del negocio, así como la definición de los procesos concretos que intervienen. Los modelos propuestos son [5]: un *modelo de valor,* que representa un conjunto de intercambios de valores y actividades realizadas por los actores del negocio; y un *modelo de procesos de negocio*, para comprender y describir los procesos de negocio relacionados con el entorno en el que se utilizará el sistema. Las notaciones utilizadas son *e3value* [8] para el primer modelo y *BPMN* [16] para el segundo.
- **Modelado de la arquitectura**. Para el modelado de la arquitectura se utilizan tres modelos: el modelo de la *arquitectura de la estructura de negocio* a nivel CIM; el modelo de *arquitectura software* a nivel PIM, que incluye una extensión del modelo anterior, pero centrado en los servicios conceptuales (nivel PIM); y el modelo de *arquitectura de servicios Web,* a nivel PSM, que contiene la información recopilada a nivel PIM y hace uso de los conceptos propios de las tecnologías de servicios Web [23]. El modelo arquitectónico de la estructura de negocio comprende todas las entidades que intervienen en un negocio y las relaciones que se establecen entre ellos (contratos). Todos los modelos se representan mediante *diagramas de clases UML extendidos* [12].
- **Modelado del comportamiento**. Los modelos arquitectónicos permiten identificar la estructura del software a implementar. Sin embargo, el comportamiento específico necesita ser modelado independientemente. El método presentado en este trabajo define tres modelos para este fin [5]: un *modelo de composición de servicios* y un *modelo de composición de servicio extendido* a nivel PIM, que representan, mediante diagramas de actividades UML, el flujo de trabajo necesario para llevar a cabo un servicio; y, a nivel PSM, varios *modelos de interfaces de servicios Web* para describir la interfaz de los servicios web implementados en el sistema.

3 Alineación negocio-tecnología: aplicación a un caso de estudio

En esta sección se presenta, ilustrado con su aplicación a un caso de estudio, el conjunto de pasos que contiene el proceso de alineación propuesto. Por motivos de espacio únicamente se reflejarán los modelos del caso de estudio correspondientes a la arquitectura (de la estructura del negocio, conceptual software y de servicios Web).

El caso de estudio que hemos utilizado para validar el proceso es el desarrollo de un **sistema para la gestión de los percentiles de crecimiento** de niños que busca mejorar el seguimiento de su crecimiento tanto por padres como por facultativos médicos. Consta de una aplicación móvil dirigida a *padres*, que permite la consulta de tablas de crecimiento o calendarios de vacunaciones, y una aplicación de escritorio para *pediatras*, que pueden enviar notificaciones o solicitar la sincronización de los datos de percentiles para un mejor seguimiento. Tanto la aplicación móvil como la de escritorio se comunican a través de un servidor radicado en el *hospital*.

3.2 Paso 1: obtención de la arquitectura de la estructura de negocio

El primer paso del método propuesto es proporcionar una representación completa de la situación de negocio del caso de estudio. Para ello se empieza modelando los servicios y valores que se identifican en el ámbito de la actividad empresarial del negocio mediante un *modelo de valor* (no incluido por motivos de espacio). Para obtenerlo fue necesario hablar directamente con los profesionales sanitarios que nos indicaron las necesidades de negocio a cubrir. Por ejemplo, desde el punto de vista del *pediatra*, la posibilidad de sincronizar los datos de los diferentes niños o, desde el punto de vista de los *padres*, visualizar gráficamente los percentiles de crecimiento. Estas necesidades de negocio nos permitieron identificar los servicios (*actividades de negocio*) que el sistema de información debía proveer tanto a *pediatras* como a *padres* (identificados como actores del negocio). Junto a este modelo de valor, también propusimos modelar las características de cada uno de los procesos que intervienen en este escenario de negocio. Esto se hizo mediante la creación del *modelo de proceso de negocio* en BPMN.

El primer conjunto de transformaciones de modelos que define nuestro proceso se refiere al objetivo de obtener un *modelo de la arquitectura de la estructura de negocio*. Esto se realizó tomando como fuente el modelo de valor y los modelos de procesos de negocio. El resultado es el modelo que se muestra en la Fig. 2.

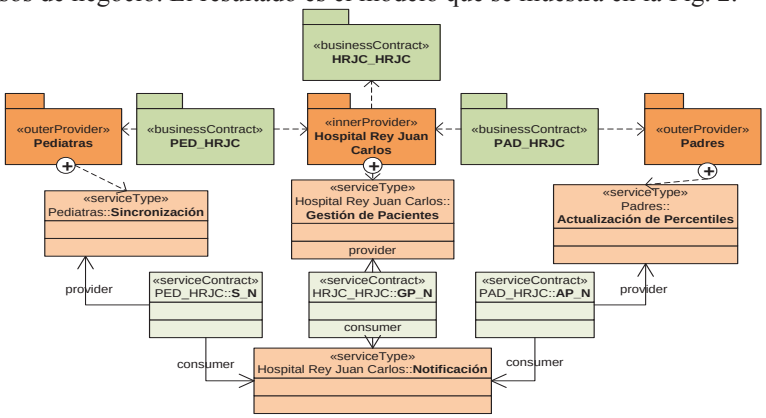

Fig. 2. Modelo de la arquitectura de la estructura de negocio.

Las entidades de negocio (*padres* y *pediatras*) se han convertido en proveedores de servicio (*service providers*). El elemento arquitectónico que actúa como conector dentro de este modelo es el contrato de negocio (*business contract*). Se obtendrá este elemento siempre que se intercambie un objeto de valor entre dos actores en el modelo de valor. Aunque los proveedores de servicios son las entidades que adquieren la responsabilidad de la situación empresarial, los que realizan realmente las actividades de negocio son los *servicios*. Como tal, las actividades de valor serán transformadas directamente en servicios. Para ilustrar la importancia de haber definido el modelo de procesos de negocio, podemos comentar cómo una actividad de "actualización de percentiles" existente en dicho modelo se utiliza como fuente para incluir en el modelo arquitectónico un servicio concreto que se encarga de llevar a cabo sus tareas.

3.3 Paso 2: obtención de la arquitectura software

Una vez que la estructura del negocio se ha modelado, nuestro método facilita la progresión en el desarrollo del sistema tomando como entrada el modelo de arquitectura de la estructura de negocio y un conjunto de reglas de transformación para obtener la arquitectura software. En este paso también se crea el modelo de composición de servicios del aspecto de comportamiento. El modelo de arquitectura software del caso de estudio se muestra en la Fig. 3 y se ha obtenido mediante la aplicación de las reglas de transformación asociadas al método (al igual que el desarrollo del aspecto de comportamiento, no se han incluido por razones de espacio).

Los elementos centrales de este modelo representan la estructura conceptual del software a construir. El origen de cada una de ellas es el siguiente: los servicios y tipos de servicio se obtienen a partir de los servicios de negocios del modelo de arquitectura de la estructura de negocio. Para el caso de estudio, el servicio que solicita la información registrada por los padres en su aplicación móvil se ha transformado directamente en un servicio software (*PAD_FrontEnd)*.

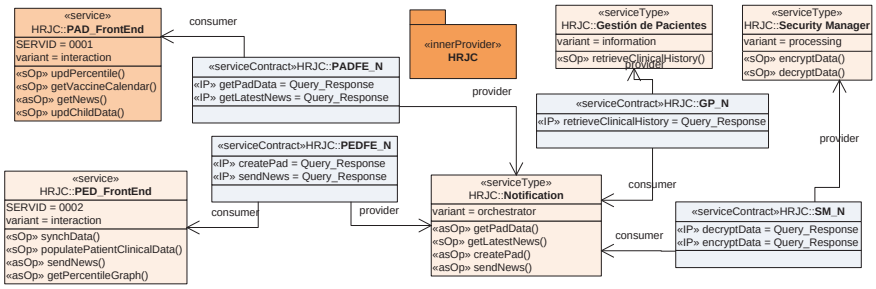

Fig. 3. Modelo de la arquitectura software.

El modelo mostrado en la Fig. 3 también contiene otros servicios que no se obtienen directamente de la arquitectura de negocios (*SecurityManager*), pero que dependen del diseño arquitectónico decidido por los desarrolladores. Al igual que en el caso de los servicios, se han definido transformaciones para especificar las relaciones entre los diferentes servicios conceptuales (contratos de servicios), las operaciones de los servicios (como unidades de funcionalidad individuales asociados a cada servicio, que en este caso obtenemos del modelo de composición de servicios que a su vez lo obtiene del modelo de procesos de negocio) y los patrones de intercambio de mensajes entre servicios.

3.4 Paso 3: obtención de la arquitectura de servicios Web

El modelo arquitectónico final es el que representa los elementos que se implementarán como artefactos de software y de acuerdo a las particularidades de las tecnologías de servicios Web. Los elementos arquitectónicos conceptuales se transformarán en entidades de las que será posible generar fragmentos de código e interfaces de servicio. El modelo de arquitectura de servicios Web resultante para el caso de estudio es el que se muestra en la Fig. 4.

Las reglas de transformación aplicadas para obtener el modelo de la Fig. 4 incluyen la definición del *servicio Web* como elemento arquitectónico principal. Así, el servicio conceptual de que realiza la sincronización de percentiles (*PED_FrontEnd)* que está alineada con la actividad de valor modelada inicialmente a nivel de negocio se transforma automáticamente en un servicio Web. Otros elementos que se obtienen semi-automáticamente son, por ejemplo, los recursos gestionados por los servicios (*resources*), derivados del tipo de servicio definido a nivel conceptual (procesamiento, información, interacción, etc.) o las operaciones de servicio (*service operations*), como funcionalidades atómicas de un servicio derivadas directamente de las operaciones de servicio definidas en el nivel de la arquitectura conceptual.

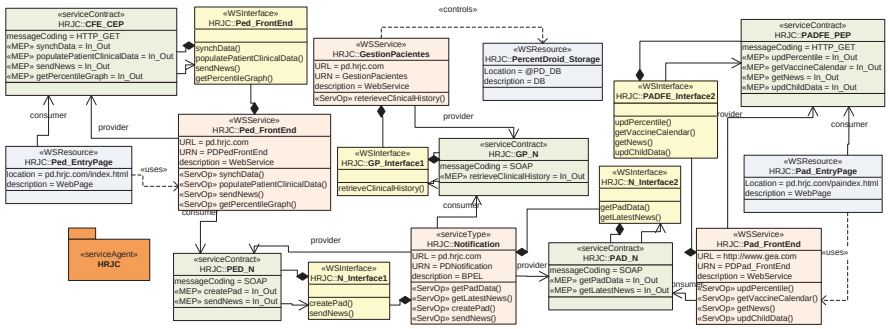

Fig. 4. Modelo de la arquitectura de Servicios Web.

3.5 Lecciones aprendidas

La primera lección obtenida de la aplicación del método propuesto al caso de estudio presentado es que el uso de un modelo de arquitectura de la estructura de negocio aporta claridad al problema de la alineación entre el negocio y la tecnología de software. Los modelos de alto nivel definidos contienen elementos que se pueden transformar fácilmente en elementos de una arquitectura software. Tras la aplicación de la propuesta hemos corroborado que la alineación es mucho más sencilla, ya que el problema puede reducirse a una alineación entre modelos arquitectónicos.

En este sentido, la decisión de utilizar un enfoque orientado a servicios para el desarrollo de software favorece asimismo la alineación del negocio con el software. Es más fácil alinear un negocio con una arquitectura orientada a servicios que con arquitecturas siguiendo otros paradigmas. La orientación a servicios cuenta con características particulares que están mucho más cerca de la idea de negocio con lo que es posible representar las infraestructuras software [11]. Tal como se afirma en [18], la orientación a servicios crea abstracciones de nivel de servicio que se corresponden con la forma en que un negocio funciona realmente.

Finalmente, el uso de transformaciones de modelos hace explícita la alineación negocio-software. La definición de transformaciones entre los elementos de diferentes modelos añade otras ventajas importantes a este proceso, tales como la capacidad de manejar las relaciones de trazabilidad entre los procesos de negocio de alto nivel y los sistemas software orientados a servicios que los soportan [21].

4 Trabajos relacionados

Con respecto a las propuestas que abordan el desarrollo de soluciones orientadas a servicios (basado en modelos o de otra manera), encontramos muchos de ellos dentro del *Programa Marco Europeo*. Proyectos como *SeCSE* [1] abordan el modelado y desarrollo de soluciones orientadas a servicios, pero sin tener en cuenta el modelado arquitectónico como una aspecto central del proceso de desarrollo. Otros proyectos como *SENSORIA* [9] o *PLASTIC* [20] consideran los servicios de modelado a nivel tecnológico, pero de nuevo consideran la arquitectura como mero artefacto a modelar.

Relacionados con el desarrollo dirigido por modelos de software orientado a servicios también encontramos el caso de *Zhang et al*. [24], que, al igual que la propuesta de *SCA* [17], considera el modelado de arquitecturas de software basadas en componentes de servicios como entidades de modelado de primer nivel, pero sin tener en cuenta los procesos de negocio o sus características. *Perovich et al.* [19] proponen una metodología de desarrollo que se inicia con la definición de modelos CIM para la arquitectura hasta llegar finalmente a niveles cercanos al código. Esta propuesta, sin embargo, no incluye soporte para modelos de comportamiento o transformaciones.

Finalmente, nos gustaría comentar dos iniciativas del ámbito comercial que abordan el desarrollo de una solución tecnológica a partir de la descripción del negocio y centrado en la arquitectura. Por un lado, Motion Lite [13], que permite el desarrollo de servicios Web pero que, a pesar de que Microsoft afirma que el proceso definido es trazable y "parcialmente automatizado", no hay ninguna referencia clara en cuanto a cuáles son sus modelos o las reglas de transformación. Por otra parte, IBM ha definido CBM (*Component Business Model*) para definir "transformaciones empresariales, dando prioridad a los objetivos estratégicos y su vinculación a las soluciones a través de las aplicaciones tradicionales o soluciones SOA" [25].

5 Conclusiones y líneas de trabajo futuras

En este trabajo hemos presentado un método centrado en la arquitectura y dirigido por modelos con el fin de abordar el reto de alinear los procesos de negocio con soluciones de software mediante la orientación a servicios. El hecho de considerar la arquitectura como artefacto central del proceso de alineación nos permite reconocer, muy temprano en el proceso de desarrollo, los elementos clave de la estructura del negocio así como sus interacciones utilizando el concepto de servicio. El uso de un enfoque basado en modelos para este proceso reduce la complejidad de alineación ya que permite que los elementos de negocio especificados en modelos de negocio sean convertidos en elementos cada vez más detallados y cercanos a la arquitectura del software objetivo. De esta forma se transfiere la información estructural del ámbito empresarial al soporte tecnológico a través de los modelos arquitectónicos orientados a servicios. En este sentido, una contribución adicional de este trabajo es la definición de un nuevo modelo de negocio. Este modelo ofrece una nueva visión de negocio que complementa otros modelos de negocios bien conocidos, tales como el modelo de valor o el modelo de proceso de negocio.

El proceso propuesto ha sido refinado y validado mediante su aplicación a un caso real del ámbito sanitario, cuyo objetivo es ayudar a la gestión de la información de

crecimiento de pacientes pediátricos. A pesar de todos los beneficios y características de nuestra propuesta, estamos trabajando en varias direcciones asociadas al proceso definido. Por un lado, estamos trabajando para completar el desarrollo de un conjunto de herramientas de modelado que soporte el método centrado en la arquitectura propuesto. Por otro lado, también nos gustaría resaltar que en estos momentos estamos aplicando el método que se presenta con algunos otros casos de estudio más complejos en el ámbito de los servicios para las ciudades inteligentes.

Agradecimientos

Esta investigación ha sido parcialmente financiada por el proyecto MASAI (TIN2011-22617/TIN) del Ministerio de Educación, Cultura y Deporte y la Red Temática Científico-Tecnológica en Ciencias de los Servicios (TIN2011-15497-E) financiado por el Ministerio de Economía y Competitividad de España. También nos gustaría dar las gracias a la Dra. Miriam Herrero y a la Dra. Erika Jiménez, del Hospital Rey Juan Carlos, por su colaboración en la definición del caso de estudio.

Referencias

[1]. ATOS, SeCSE methodology, version 3, IST European integrated project SeCSE, in: Proceedings of Sixth Framework Programme, Deliverable A5.D4.2.
[2]. Aversano, L., Grasso, C., Tortorella, M., A Literature Review of Business/IT Alignment Strategies, Procedia Technology, 5, pp. 462-474, ISSN 2212-0173.
[3]. Broy M. Model Driven, Architecture-Centric Modeling in Software Development. In Proc. of 9th Intl. Conf. in Eng. Complex Comp. Syst., pp. 3-12, IEEE Computer Society, 2004.
[4]. Cherbakov, I., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G, 2005. Impact of Service Orientation at Business Level. IBM Systems Journal, vol. 44, issue 4, pp. 653-668.
[5]. De Castro, V., Marcos, E., Vara, J. M. Applying CIM-to-PIM model transformations for the service-oriented development of information systems. IST 53(1): 87-105. 2011.
[6]. De Castro, V., Marcos, E., 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 J on Coop. Inf. Syst.. 18(2): 225-260.
[7]. Glissmann S., Sanz, J.L.C., 2010. Business Architectures for the Design of Enterprise Service Systems. Handbook of Service Science, Springer.
[8]. Gordijn, J., Akkermans, J.M., 2003. Value based requirements engineering: exploring innovative e-commerce idea, Requirements Engineering Journal 8 (2) 114–134.
[9]. J.L. Fiadeiro, A. Lopes, L. Bocchi, The SENSORIA Reference Modelling Language: Primitives for Service Description, available at www.sensoria-ist.edu, 2006.
[10]. Karakostas, B., Y. Zorgios, 2008. Engineering Service Oriented Systems: A Model Driven Approach. IGI Global, Hershey.
[11]. Krafzig, D., Banke, K., Slama, D., 2004. Enterprise SOA Service Oriented Architecture Best Practices, Prentice Hall PTR, Upper Saddle River.
[12]. Marcos, E., De Castro, V., Vela, B., 2003. Representing web services with UML: a case study. 1st Int. Conf. on Service Oriented Computing, LNCS 2910, 2003, pp. 17–27.
[13]. Merrifield R., Tobey, J., 2006. Motion Lite: A Rapid Application of the Business Architecture Techniques Used by Microsoft Motion. Microsoft.
[14]. Nayak, N., Linehan, M. H., Nigam, A., et al., 2007. Core business architecture for a service-oriented enterprise. IBM Systems Journal 46(4): 723-742.

[15]. OMG, 2001. Model Driven Architecture (MDA), Document No. ormsc/2001-07-01, available at: http://www.omg.com/mda.
[16]. OMG, 2008. Business Process Modelling Notation, Version 1.1, http://www.bpmn.org/Documents/BPMN1-1Specification.pdf
[17]. Open Service Oriented Architecture Collaboration, Service Component Architecture Project. http://www.osoa.org/ , 2007.
[18]. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F., 2008. Service-Oriented Computing. Research Roadmap. Int. J. of Coop. Inf. Syst., 17 (2), pp. 223–255.
[19]. Perovich, D., Bastarrica, M.C., Rojas, C., 2009. Model-driven approach to software architecture design. Proc. of the 4th Int. Workshop on Sharing and Reusing Architectural Knowledge, SHARK'09, IEEE Computer Society, Vancouver, Canada, 2009, pp. 1–8.
[20]. PLASTIC Project. (2006). D1.2: Formal description of the PLASTIC conceptual model and of its relationship with the PLASTIC platform toolset. http://www-c.inria.fr/plastic
[21]. Santiago, I., Jiménez, A., Vara, J.M., De Castro, V., Bollati, V., Marcos, E., 2012. Model-Driven Engineering as a new landscape for traceability management: A systematic literature review. Information & Software Technology 54(12): 1340-1356.
[22]. Sanz, J.L.C., Ren, G., Glissmann, S., Spohrer, J., Leung, Y.T., 2010. What is Business Architecture and how can it contribute to Service Science. Frontiers in Service Conference, Karlstad, Sweden.
[23]. W3C, 2004. Web Services Architecture (WSA). http://www.w3.org/TR/ws-arch/.
[24]. Zhang T., S. Ying, S. Cao, X. Jia. A Modeling Framework for Service-Oriented Architecture, in: Proceedings of the Sixth International Conference on Quality Software (QSIC 2006), pp. 219-226
[25]. Zhang, L.-J., Zhou, N., Chee, Y.-M., Jalaldeen, A., Ponnalagu, K., Sindhgatta, R., Arsanjani, A., Bernardini, F., 2008.SOMA-ME: A platform for the model-driven design of SOA solutions. IBM Systems Journal 47(3): 397-413.

Generating reliable services' composition using A-policies: a model-driven approach*

Genoveva Vargas-Solar[1], Valeria de Castro[4], Plácido Antonio de Souza Neto[5,6], Javier A. Espinosa-Oviedo[3], Esperanza Marcos[4], Martin A. Musicante[5], José-Luis Zechinelli-Martini[2,1], and Christine Collet[3]

[1] French Council of Scientific Research, LIG-LAFMIA, Grenoble, France
Genoveva.Vargas-Solar@imag.fr
[2] Fundación Universidad de las Américas, San Andrés Cholula, Puebla, México
joseluis.zechinelli@udlap.mx
[3] Grenoble Institute of Technology, Grenoble, France
Christine.Collet@grenoble-inp.fr, Javier.Espinosa@imag.fr
[4] Universidad Rey Juan Carlos, Móstoles, Spain
Valeria.deCastro@urjc.es, esperanza.marcos@urjc.es
[5] DIMAp - UFRN, ForAll - Formal Methods and Language Research Laboratory, Natal - RN, Brasil
mam@dimap.ufrn.br
[6] Instituto Federal do Rio Grande do Norte, Natal - RN, Brasil
placido.neto@ifrn.edu.br

Abstract. This paper presents an approach for modeling and associating *A-Policies* to services' based applications. It proposes to extend the SOD-M model driven method with (i) the π-SCM an *A-Policy* services' composition meta-model for representing non-functional constraints associated to services' based applications; (ii) the π-Pews meta-model providing guidelines for expressing the composition and the policies; and, (iii) model to model and model to text transformation rules for semi-automatizing the implementation of reliable services' compositions.

Keywords: MDD, Service Oriented Applications, Non-functional Properties

1 Introduction

Service oriented computing is at the origin of an evolution in the field of software development. An important challenge of service oriented development is to ensure the alignment between IT systems and the business logic. Thus, organizations are seeking for mechanisms to deal with the gap between the systems developed and business needs [2]. The literature stresses the need for methodologies and techniques for service oriented analysis and design, claiming that they

* This work is partially financed by the projects Orchestra ECOS-ANUIES, Clever STIC-AMSUD, Masai, National Scientific and Technological Network of Service Science and CAPES/STIC-AMSUD Brasil, BEX 4112/11-3.

are the cornerstone in the development of meaningful services' based applications [13]. In this context, some authors argue that the convergence of model-driven software development, service orientation and better techniques for documenting and improving business processes are the key to make real the idea of rapid, accurate development of software that serves, rather than dictates, software users' goals [19].

Service oriented development methodologies providing models, best practices, and reference architectures to build services' based applications mainly address functional aspects [1, 6, 18, 14]. Non-functional aspects concerning services' and application's "semantics", often expressed as requirements and constraints in general purpose methodologies, are not fully considered or they are added once the application has been implemented in order to ensure some level of reliability (e.g., data privacy, exception handling, atomicity, data persistence). This leads to services' based applications that are partially specified and that are thereby partially compliant with application requirements.

The objective of this work is to model non-functional constraints and associate them to services' based applications early during the services' composition modeling phase. Therefore this paper presents π-SOD-M, a model-driven method that extends the SOD-M [18] for building reliable services' based information systems. Our work proposes to extend the SOD-M [18] method with (i) the notion of *A-Policy* [10] for representing non-functional constraints associated to services' based applications. We also (ii) defines the π-PEWS meta-model [17] providing guidelines for expressing the composition and the *A-policies*. Finally, our work (iii) defines model to model transformation rules for generating the π-PEWS model of a reliable services' composition starting from the extended services' composition model; and, model to text transformations for generating the corresponding implementation.

The remainder of the paper is organized as follows. Section 2 gives an overview of our approach. It describes a motivation example that integrates and synchronizes well-known social networks services namely Facebook, Twitter and, Spotify. Sections 3, 4, and 5 describe respectively the three key elements of our proposal, namelly the π-SCM and π-PEWS meta-models and the transformation rules that support the semi-automatic generation of reliable services' compositions. Section 6 analyses related work concerning policy/contract based programming and, services' composition platforms. Section 7 concludes the paper and discusses future work.

2 Modeling reliable services' compositions with π-SOD-M

Consider for instance the following scenario. An organization wants to provide the services' based application "To Publish Music" that monitors the music a person is listening during some periods of time and sends the song title to this person's Twitter and Facebook accounts. Thus, this social network user will have her status synchronized in Twitter and Facebook (i.e., either the same title is

published in both accounts or it is not updated) with the title of the music she is listening in Spotify. For developing this services' based application it is necessary to compose the following services calling their exported methods: a) The music service Spotify exports a method for obtaining information about the music a given user is listening: get-Last-Song (userid): String ; b) Facebook and Twitter services export a method for updating the status of a given user update-Status (userid, new-status): String.

The "To Publish Music" scenario starts by contacting the music service Spotify for retrieving the user's musical status (activity Get Song). Twitter and Facebook services are then contacted in parallel for updating the user's status with the corresponding song title (activities Update Twitter and Update Facebook).

Given a set of services with their exported methods known in advance or provided by a service directory, building services' based applications can be a simple task that implies expressing an application logic as a services' composition. The challenge being ensuring the compliance between the specification and the resulting application. Software engineering methods (e.g., [1, 6, 18, 14]) today can help to ensure this compliance, particularly when information systems include several sometimes complex business processes calling Web services or legacy applications exported as services.

2.1 Modeling a services' based application

Figure 1 shows SOD-M that defines a service oriented approach providing a set of guidelines to build services' based information systems (SIS) [18]. Therefore, SOD-M proposes to use services as first-class objects for the whole process of the SIS development and it follows a Model Driven Architecture (MDA) [15] approach. Extending from the highest level of abstraction of the MDA, SOD-M provides a conceptual structure to: first, capture the system requirements and specification in high-level abstraction models (computation independent models, CIMs); next, starting from such models build platform independent models (PIMs) specifying the system details; next transform such models into platform specific models (PSMs) that bundles the specification of the system with the details of the targeted platform; and finally, serialize such model into the working-code that implements the system. As shown in Figure 1, the SOD-M model-driven process begins by building the high-level CIMs and enables specific models PSMs for a service platform to be obtained as a result [18]. Referring to the "To Publish Music" application, using SOD-M the designer starts defining an E3value model [7] at the CIM level and then the corresponding models of the PIM are generated leading to a services' composition model (SCM).

Now, consider that besides the services' composition that represents the order in which the services are called for implementing the application "To Publish Music" it is necessary to model other requirements that represent the (i) conditions imposed by services for being contacted, for example the fact the Facebook

[7] The E3 value model is a business model that represents a business case and allows to understand the environment in which the services' composition will be placed [9].

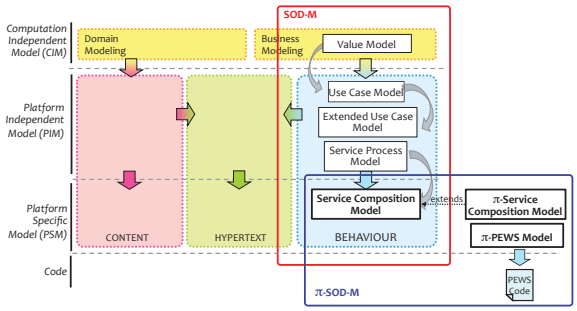

Fig. 1: SOD-M development process

and Twitter require authentication protocol in order to call their methods for updating the wall; (ii) the conditions stemming from the business rules of the application logic, for example the fact that the walls in Facebook and Twitter must show the same song title and if this is not possible then none of them is updated.

2.2 Modeling non-functional constraints of services' based applications

Adding non-functional requirements and services constraints in the services' composition is a complex task that implies programming protocols for instance authentication protocols to call Facebook and Twitter in our example, and atomicity (exception handling and recovery) for ensuring a true synchronization of the song title disseminated in the walls of the user's Facebook and Twitter accounts.

Service oriented computing promotes ease of information systems' construction thanks, for instance, to services' reuse. Yet, this is not applied to non-functional constraints as the ones described previously, because they do not follow in general the same service oriented principle and because they are often not fully considered in the specification process of existing services' oriented development methods. Rather, they are either supposed to be ensured by the underlying execution platform, or they are programmed through ad-hoc protocols.

Our work extends SOD-M for building applications by modeling the application logic and its associated non-functional constraints and thereby ensuring the generation of reliable services' composition. As a first step in our approach, and for the sake of simplicity we started modeling non-functional constraints at the PSM level. Thus, in this paper we propose the π-SCM, the services' composition meta-model extended with *A-policies* for modeling non-functional constraints (highlighted in Figure 1 and described in Section 3). π-SOD-M defines the π-PEWS meta-model providing guidelines for expressing the services' composition and the *A-policies* (see Section 4), and also defines model to model transformation rules for generating π-PEWS models starting from π-SCM models that will support executable code generation (see Section 5).

3 π services' composition meta-model

The *A-policy* based services' composition meta-model (see in Figure 2) represents a workflow needed to implement a services' composition, identifying those entities that collaborate in the business processes (called BUSINESS COLLABORATORS [8]) and the ACTIONS that they perform. This model is represented by means of a UML activity diagram. This model includes the typical modeling elements of the activity diagram such as ACTIVITYNODES, INITIALNODES and FINALNODES, DECISIONNODES, etc., along with new elements defined by SOD-M such as BUSINESS COLLABORATORS, SERVICEACTIVITY and ACTION. A BUSINESS COLLABORATOR element represents those entities that collaborate in the business processes by performing some of the required actions. They are graphically presented as a partition in the activity diagram. ACTION, a kind of EXECUTABLENODE, are represented in the model as an activity. There are two types of actions: i) a WebService (attribute Type is WS); and ii) a simple operation that is not supported by a Web Service, called an ACTIVITYOPERATION (attribute Type is AOP). The SERVICEACTIVITY element is a composed activity that must be carried out as part of a business service and is composed of one or more executable nodes.

Figure 2 illustrate the service compostion model of the "To Publish Music" scenario. There are three external business collaborators (*Spotify, Twitter* and *Facebook* [9]). It also shows the business process of the "To Publish Music" application that consists of three service activities: *Listen Music*, *Public Music* and *Confirmation*. Note that the action *Publish Music* of the application calls the actions of two service collaborators namely *Facebook* and *Twitter*.

Instead of programming different protocols within the application logic, we propose to include the modeling of non-functional constraints like transactional behaviour, security and adaptability at the early stages of the services' composition engineering process. We model non-functional constraints of services' compositions using the notion of *A-policy* [10, 5], a kind of pattern for specifying *A-policy* types. In order to represent constraints associated to services compositions, we extended the SOD-M services' composition model with two concepts: RULE and A-POLICY.

The RULE element represents an event - condition - action rule where the EVENT part represents the moment in which a constraint can be evaluated according to a condition represented by the CONDITION part and the action to be executed for reinforcing it represented by the ACTION part.

An *A-policy* groups a set of rules. It describes global variables and operations that can be shared by the rules and that can be used for expressing their Event and Condition parts. An *A-Policy* is associated to the elements BUSINESSCOLLABORATOR, SERVICEACTIVITY and, ACTION of the π-SCM meta-model.

[8] We use CAPITALS for referring to meta-models' classes.

[9] We use *italics* to refer to concrete values of the classes of a model that are derived from the classes of a meta-model.

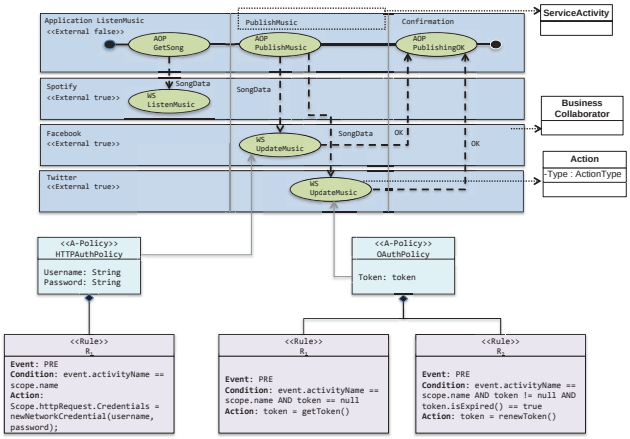

Fig. 2: Services' composition model for the business service "To publish music"

Given that *Facebook* and *Twitter* services require authentication protocols in order to execute methods that will read and update the users' space. A call to such services must be part of the authentication protocol required by these services. In the example we associate two authentication policies, one for the open authentication protocol, represented by the class Twitter OAuthPolicy that will be associated to the activity UpdateTwitter (see Figure 2). In the same way, the class Facebook HTTPAuthPolicy, for the http authentication protocol will be associated to the activity UpdateFacebook. OAuth implements the open authentication protocol. As shown in Figure 2, the *A-policy* has a variable Token that will be used to store the authentication token provided by the service. This variable type is imported through the library OAuth.Token. The *A-policy* defines two rules, both can be triggered by events of type ActivityPrepared: (i) if no token has been associated to the variable token, stated in by the condition of rule R_1, then a token is obtained (action part of R_1); (ii) if the token has expired, stated in the condition of rule R_2, then it is renewed (action part of R_2). Note that the code in the actions profits from the imported OAuth.Token for transparently obtaining or renewing a token from a third party.

HTTP-Auth implements the HTTP-Auth protocol. The *A-policy* imports an http protocol library and it has two variables username and password. The event of type ActivityPrepared is the triggering event of the rule R_1. On the notification of an event of that type, a credential is obtained using the username and password values. The object storing the credential is associated to the scope, i.e., the activity that will then use it for executing the method call.

Thanks to rules and policies it is possible to model and associate non-functional properties to services' compositions and then generate the code. For example, the atomic integration of information retrieved from different social network services or for providing security in the communication channel when the payment service is called.

Back to the definition process of a SIS, once the *A-policy* based services' composition model has been defined, then it can be transformed into a model (i.e., π-PEWS model) that can support then executable code generation. The following Section describes the π-PEWS meta-model that supports this representation.

4 π-Pews meta-model

The idea of the π-Pews meta-model is based on the services' composition approach provided by the language PEWS[3, 17] (*Path Expressions for Web Services*), a programming language that lets the service designer combine the methods or subprograms that implement each operation of a service, in order to achieve the desired application logic. Figure 3 presents the π-Pews meta-model.

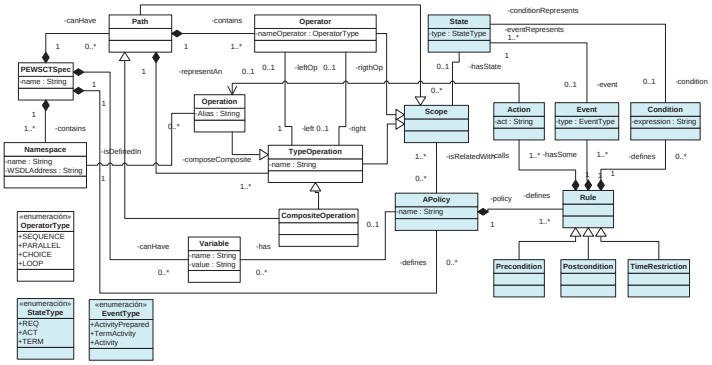

Fig. 3: π-Pews Metamodel

Namespace represents the interface exported by a service, Operation that represents a call to a service method, CompositeOperation, and Operator for representing a services' composition and Path representing a services' composition. A Path can be an Operation or a Compound Operation denoted by an identifier. A Compound Operation is defined using an Operator that can represent sequential and parallel composition of services, choice among services, the sequential and parallel repetition of an operation, or the conditional execution of an operation.

A-Policies that can be associated to a services' composition: A-Policy, Rule, Event, Condition, Action, State, and Scope.

As shown in the diagram an A-Policy is applied to a Scope that can be either an Operation (e.g., an authentication protocol associated to a method exported by a service), an Operator (e.g., a temporal constraint associated to a sequence of operators, the authorized delay between reading a song title in Spotify and updating the walls must be less then 30 seconds), and a Path (e.g.,

executing the walls' update under a strict atomicity protocol - all or noting). It groups a set of ECA rules, each rule having a classic semantics, i.e, *when an event of type E occurs if condition C is verified then execute the action A*. Thus, an *A-policy* represents a set of reactions to be possibly executed if one or several triggering events of its rules are notified.

The class SCOPE represents any element of a services' composition (i.e., operation, operator, path). The class A-POLICY represents a recovery strategy implemented by ECA rules of the form EVENT - CONDITION - ACTION. A *A-policy* has variables that represent the view of the execution state of its associated scope, that is required for executing the rules. The value of a variable is represented using the type VARIABLE. The class A-POLICY is specialized for defining specific constraints, for instance authentication *A-policies*.

Given a π-SCM model of a specific services' based application, it is possible to generate its corresponding π-PEWS model thanks to transformation rules. The following Section describes the transformation rules between the π-SCM and π-PEWS meta-models of our method.

5 Transformation rules

Figure 4 shows the transformation principle between the elements of the π-SCM meta-model used for representing the services' composition into the elements of the π-PEWS meta-model. There are two groups of rules: those that transform services' composition elements of the π-SCM to π-PEWS meta-models elements; and those that transform rules grouped by policies into *A-policy* types.

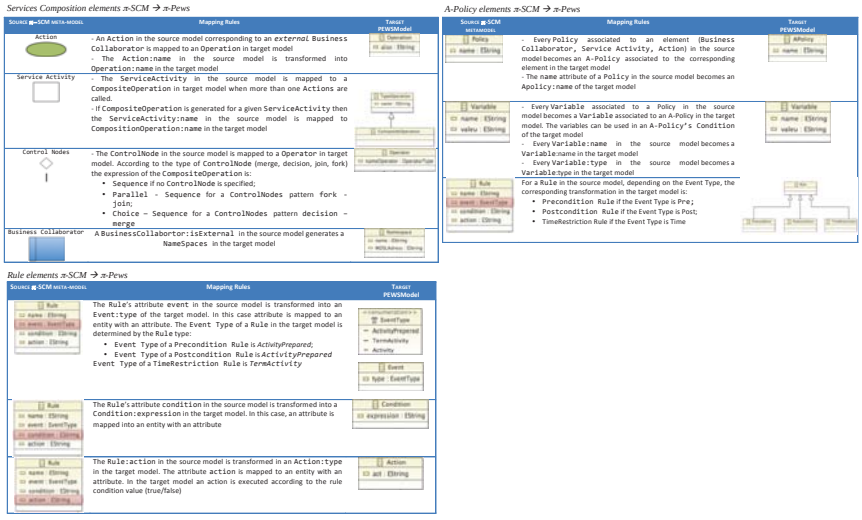

Fig. 4: π-SCM to π-PEWS transformation

6 Related works

Current standards in services' composition implement functional, non-functional constraints and communication aspects by combining different languages and protocols. The selection of the adequate protocols for adding a specific non-functional constraints to a services' composition (e.g., security, transactional behaviour and adaptability) is responsibility of a programmer. As a consequence, the development of an application based on a services' composition is a complex and a time-consuming process. This is opposed to the philosophy of services that aims at facilitating the integration of distributed applications.

Software process methodologies for building services based applications have been proposed in[14, 16, 7, 4], and they focus mainly on the modeling and construction process of services based business processes that represent the application logic of information systems. *Design by Contract* [8] is an approach for specifying web services and verifying them through runtime checkers before they are deployed. A contract adds behavioral information to a service specification, that is, it specifies the conditions in which methods exported by a service can be called. Contracts are expressed using the language *jmlrac* [12]. The *Contract Definition Language* (CDL) [16] is a XML-based description language, for defining contracts for services. There are an associated architecture framework, design standards and a methodology, for developing applications using services. A services' based application specification is generated after several [11] B-machines refinements that describe the services and their compositions.

As WS-* and similar approaches, our work enables the specification and programing of crosscutting aspects (i.e., atomicity, security, exception handling, persistence). In contrast to these approaches, our work specifies policies for a services' composition in an orthogonal way. Besides, these approaches suppose that non-functional requirements are implemented according a the knowledge that a programmer has of a specific application requirements but they are not derived in a methodological way, leading to ad-hoc solutions that can be difficult to reuse. In our approach, once defined *A-Policies* for a given application they can be reused and/or specialized for another one with the same requirements or that uses services that impose the same constraints.

7 Conclusions and future work

This paper presented π-SOD-M for specifying and designing reliable service based applications. We model and associate policies to services' based applications that represent both systems' cross-cutting aspects and use constraints stemming from the services used for implementing them. We extended the SOD-M method, particularly the π-SCM (services' composition meta-model) and π-PEWS meta-models for representing both the application logic and its associated non-functional constraints and then generating its executable code.

Non-functional constraints are related to business rules associated to the general "semantics" of the application and in the case of services' based applications,

they also concern the use constraints imposed by the services. We are currently working on the definition of a method for explicitly expressing such properties in the early stages of the specification of services based applications. Having such business rules expressed and then translated and associated to the services' composition can help to ensure that the resulting application is compliant to the user requirements and also to the characteristics of the services it uses.

Programming non-functional properties is not an easy task, so we are defining a set of predefined *A-policy* types with the associated use rules for guiding the programmer when she associates them to a concrete application. *A-policy* type that can also serve as patterns for programming or specializing the way non-functional constraints are programmed.

References

1. A. Arsanjani, S. Ghosh, A.A.T.A.S.G.K.H.: SOMA: A method for developing service-oriented solutions. IBM System Journal **47**(3) (2008)
2. Bell, M.: Service-Oriented Modeling: Service Analysis, Design, and Architecture. (2008)
3. Cheikh, B., Mirian, H.F., Martin Alejandro, M.: Composing Web Services with PEWS: A Trace-Theoretical Approach. In: ECOWS. (2006) 65–74
4. Ervin, R., Dimitris, D., Anthony J. H., S.: A survey of service oriented development methodologies
5. Espinosa-oviedo, J.A., Vargas-Solar, G., Zechinelli-Martini, J.L., Collet, C.: Non-Functional Properties and Services Coordination Using Contracts. In: In proceedings of the 13th Int. Database Engineering and Applications Symposium (IDEAS 09), Cetraro, Italy, ACM (2009)
6. et. al, A.B.: SOA Development Using the IBM Rational Software Development Platform: A Practical Guide. In: Rational Software. (2005)
7. George, F., Sooksathit, M.: Towards software development methodology for web services. In: SoMeT. (2005) 263–277
8. Heckel, R., Lohmann, M.: Towards contract-based testing of web services. In Pezzé, M., ed.: Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS 2004). Volume 116. (2005) 145–156
9. J., Gordijn, J.A.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Engineering Journal **8**(2) (2003)
10. Javier-Alfonso, E.O., Genoveva, V.S., JosÓ-Luis, Z.M., Christine, C.: Policy driven services coordination for building social networks based applications. In: In Proc. of the 8th Int. Conference on Services Computing (SCC'11), Work-in-Progress Track, Washington, DC, USA, IEEE (July 2011)
11. Jean-Raymond, A., Matthew K. O., L., David, N., P. N., S., Ib Holm, S.: The b-method. In Prehn, S., Toetenel, W.J., eds.: VDM Europe (2). Volume 552 of Lecture Notes in Computer Science., Springer (1991) 398–405
12. Leavens, G.T., Cheon, Y., Clifton, C., Ruby, C., Cok, D.R.: How the design of jml accomodates both runtime assertion checking and formal verification. In: FMCO. (2002) 262–284
13. M. Papazoglou, P. Traverso, S.D.F.L.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer **40**(11) (2007)
14. Mike P., P., Willem-Jan, v.d.H.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. **2**(4) (2006) 412–442

15. Miller, J., Mukerji, J.: Mda guide. (2003)
16. Nikola, M.: Contract-based Web Service Composition. PhD thesis, Humboldt-UniversitÃd't zu Berlin (2006)
17. Plácido A., S.N., Martin Alejandro, M., Genoveva, V.S., José-Luis, Z.M.: Adding Contracts to a Web Service Composition Language. LTPD - 4th Workshop on Languages and Tools for Multithreaded, Parallel and Distributed Programming (September 2010)
18. V. De Castro, E. Marcos, R.W.: Towards a service-oriented mda-based approach to the alignment of business processes with it systems: From the business model to a web service composition model. International Journal of Cooperative Information Systems **18**(2) (2009)
19. Watson, A.: A brief history of MDA (2008)

Composition and Self-Adaptation of Service-Based Systems with Feature Models*

Javier Cubo, Nadia Gamez, Lidia Fuentes, Ernesto Pimentel

Universidad de Málaga, Dpto de Lenguajes y Ciencias de la Computación
{cubo,nadia,lff,ernesto}@lcc.uma.es

Summary of the contribution

The adoption of mechanisms for reusing software in pervasive systems has not yet become standard practice. This is because the use of pre-existing software requires the selection, composition and adaptation of prefabricated software parts, as well as the management of some complex problems such as guaranteeing high levels of efficiency and safety in critical domains. In addition to the wide variety of services, pervasive systems are composed of many networked heterogeneous devices with embedded software. In this work, we promote the safe reuse of services in service-based systems using two complementary technologies, Service-Oriented Architecture and Software Product Lines. In order to do this, we extend both the service discovery and composition processes defined in the DAMASCo framework, which currently does not deal with the service variability that constitutes pervasive systems. We use feature models to represent the variability and to self-adapt the services during the composition in a safe way taking context changes into consideration. We illustrate our proposal with a case study related to the driving domain of an Intelligent Transportation System, handling the context information of the environment.

In this work, we present the integration process of the feature models (FM) into DAMASCo to deal with the services' variability in the composition. Firstly, in addition to the BPEL/WF descriptions, we have a FM for each service describing its variability. Then, each business process corresponds with a valid configuration of the FM that represents it. We focus on the representation of the service variability *wrt* the context. After a client executes a request, both the DAMASCo model transformation and the semantic-based service discovery process are activated. Due to the high variability of the services, it is possible that a small variation of a service could be needed. Then, a new process using FM is added to the DAMASCo framework, called service family discovery, to find a new matching as regards a certain context. A new valid configuration of that family containing this feature, representing the particular services, is automatically created, and the new service self-adaptation process added to DAMASCo is executed. Then, the CA-STS corresponding with this FM configuration is automatically created. Finally, this CA-STS interface is transformed into a WF/BPEL, composed with the services to satisfy the request.

* This work has been published in the 13th Int. Conf. on Software Reuse (ICSR 2013), LNCS, vol. 7925, 326-342 (2013). Partially supported by projects TIN2008-05932, TIN2008-01942, TIN2012-35669, TIN2012-34840 and CSD2007-0004 funded by Spanish Gov, FEDER; P09-TIC-05231, P11-TIC-7659 by Andalusian Gov; FP7-317731 by EU; Univ. de Málaga, Campus Excelencia Int. Andalucía Tech.

A service-oriented framework for developing cross cloud migratable software*

Javier Miranda[1], Juan Manuel Murillo[2], and Carlos Canal[3]

[1] Gloin, Calle de las Ocas 2, Cáceres, Spain
jmiranda@gloin.es
[2] Department of Information Technology and Telematic Systems Engineering, University of Extremadura, Spain
juanmamu@unex.es
[3] Department of Computer Science, University of Málaga, Spain
canal@lcc.uma.es

Summary of the contribution

Whilst cloud computing has burst into the current scene as a technology that allows companies to access high computing rates at limited costs, cloud vendors have rushed to provide tools that allow developers to build software for their cloud platforms. Cloud applications are developed using those tools, which provide different cloud-specific APIs, libraries, and even different project structures that vary depending on which cloud platform the software will be hosted. Consequently, applications developed with these tools are often tightly coupled to those platform's specific service implementations and restrictions. A scenario where component-based applications are developed for being deployed across several clouds, and each component can independently be deployed in one cloud or another, remains fictitious due to the complexity and the cost of their development.

This paper presents a cloud development framework that allows applications to be constructed as a composition of software components (cloud artefacts), where each component can be freely migrated between cloud platforms without having to redevelop the entire application. Information about cloud deployment and cloud integration is separated from the source code and managed by the framework. Interoperability between interdependent components deployed in different clouds is achieved by means of software adapters which automatically generate services and service clients. This allows software developers to segment their applications into different modules that can easily be deployed and redistributed across heterogeneous cloud platforms. This paper also analyzes the results of using the proposed framework in the development of an industrial research project as a validation of the approach.

* This work has been published in the Journal of Systems and Software 86(9), 2294-2308 (2013).

Una Propuesta Orientada a Servicios para la Prevención de Riesgos Personales Derivados de la Calidad del Aire

Juan Boubeta-Puig, Guadalupe Ortiz Bellot e Inmaculada Medina-Bulo

Grupo UCASE de Ingeniería del Software
Departamento de Ingeniería Informática, Universidad de Cádiz,
C/Chile 1, 11002 Cádiz, España
{juan.boubeta,guadalupe.ortiz,inmaculada.medina}@uca.es

Resumen La calidad del aire es un factor que ha tomado gran relevancia en los últimos años y que puede afectar seriamente a la salud y a la calidad de vida de los ciudadanos. Actualmente los medios que nos permiten mantenernos informados sobre la calidad del aire en general se caracterizan por no proporcionar la información en tiempo real ni mecanismos de información de fácil acceso para el ciudadano y, sobre todo, no se adaptan a las condiciones específicas de cada ciudadano particular. En este artículo proponemos la implementación de una arquitectura orientada a servicios que nos va a permitir detectar cambios en la calidad del aire en tiempo real y poner esta información a disposición del usuario en su móvil, notificándole inmediatamente de alertas personalizadas cuando se detecte algún nivel potencialmente peligroso para su salud, procurando así la prevención de riesgos personales.

Palabras clave: Procesamiento de eventos complejos, arquitecturas orientadas a servicios, calidad del aire.

1. Introducción y Motivación

Con motivo de la polución, entre otros factores, la calidad del aire juega un papel de gran relevancia cuando se trata de la salud de los ciudadanos, pudiendo propiciar o empeorar determinadas enfermedades o incluso causar la muerte de determinados grupos de riesgo [23]. En este sentido, su control es de gran importancia para una muestra representativa de la ciudadanía como, por ejemplo, las personas mayores, las personas que hacen deporte en el exterior, los niños que juegan al aire libre o las personas que tienen algún tipo de afección de corazón o de alergia a determinadas sustancias que entran en nuestro organismos a través de la inhalación. Si bien, estos son grupos de riesgo, en general la calidad del aire afecta a todos los ciudadanos en el día a día en mayor o menor medida.

Al igual que el clima, la calidad del aire cambia, no sólo de un día para otro, sino también a lo largo de un mismo día. Es por esto que es importante estar informado de la calidad del aire en tiempo real, pudiendo así prevenir riesgos personales. Hay tres factores de especial relevancia que deben tenerse en cuenta

a la hora de informar al ciudadano sobre la calidad del aire: 1) que la información se actualice en tiempo real; 2) que la información pueda llegar al ciudadano de una manera cómoda y de fácil acceso; 3) que la información se pueda personalizar en función de las características específicas de un ciudadano en particular —no es lo mismo cómo afecta la polución a un alérgico en primavera a cómo puede afectar a un joven sano sin problemas respiratorios.

Actualmente los medios existentes que nos permiten mantenernos informados sobre la calidad del aire, en general, se caracterizan por no suministrar la información en tiempo real [14,19], no proveer mecanismos de información de fácil acceso para el ciudadano [1,11] y, sobre todo, no proporcionar información particularizada para las condiciones de un ciudadano en concreto [8,12]. Por ello, en este artículo se propone que esa información llegue en tiempo real al ciudadano de una forma cómoda y sencilla y adaptándolo a sus necesidades particulares.

Para lograrlo, proponemos el desarrollo de un sistema que sea capaz de procesar diversas fuentes de información que nos permitan conocer y valorar la calidad del aire en cada instante y mantener informado al ciudadano —*sistema de detección*, de aquí en adelante. Basándonos en nuestra experiencia en otros dominios de aplicación [5,6], creemos firmemente que el uso de tecnologías emergentes como el procesamiento de eventos complejos o *Complex Event Processing* (CEP), que permite analizar diversos flujos de información y detectar, en base a patrones de eventos definidos en ese dominio, situaciones críticas en tiempo real, es clave en la implementación del sistema de detección. Por otra parte, para poder integrar en el sistema flujos de información de diversos proveedores de Internet —presumiblemente en distintos formatos— juzgamos que el sistema debe estar a su vez integrado por un bus de servicios empresarial o *Enterprise Service Bus* (ESB). Así pues, el principal objetivo de este trabajo es demostrar que CEP es una solución efectiva y alternativa para detectar en tiempo real los distintos niveles de calidad del aire en ciudades de todo el mundo.

El resto del artículo se estructura de la siguiente forma. En la Sección 2 se definen los conceptos previos que facilitan la comprensión del resto del artículo. A continuación, en la Sección 3 se detalla la arquitectura definida para llevar a cabo nuestra propuesta; seguido de la Sección 4 donde se describe un caso de estudio para la detección de niveles de calidad de aire en distintas zonas geográficas. En la Sección 5 se discute la propuesta y en la Sección 6 se enumeran algunos trabajos relacionados. Finalmente se presentan las conclusiones y el trabajo futuro.

2. Conceptos Previos

2.1. Internet de las Cosas

El Internet de las Cosas o *Internet of Things* (IoT) es un concepto clave dentro de la propuesta que aquí describimos. Se define como una red formada por objetos interconectados que pueden ser identificados unívocamente y representados en un mundo virtual que cuenta con diversos dominios de aplicación [3]. Existen múltiples plataformas IoT de donde se pueden obtener en tiempo real

datos relevantes para estos dominios. Xively [15] es una de estas plataformas, bien documentada y altamente escalable, que gestiona cada día millones de datos provenientes de miles de personas y compañías, permitiendo almacenar, compartir y extraer datos públicos y/o privados en tiempo real ofrecidos por los distintos dispositivos localizados a nivel mundial. Para ello, ofrece una API RESTful que proporciona los valores observados en formato XML, CSV o JSON.

2.2. Procesamiento de Eventos Complejos

CEP [16] es 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). 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, esto es, 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 las aplicaciones o usuarios finales. La característica principal de estos eventos complejos procesados con CEP es que pueden ser identificados e informados en tiempo real, respondiendo así a situaciones que cambien rápidamente. De este modo se consigue una mayor reducción de la latencia en la toma de decisiones frente a la lograda mediante el uso del software tradicional de análisis de eventos.

2.3. Bus de Servicios Empresarial y SOA 2.0

Un ESB es un *middleware* que permite integrar diferentes aplicaciones y servicios de entornos heterogéneos proporcionando interoperabilidad entre los distintos protocolos de comunicación [10], permitiendo además la exposición de las distintas aplicaciones como servicios. Entre las múltiples ventajas que proporcionan los ESB, destacan la transparencia de la ubicación, conversión de protocolos de transporte, transformación, enrutamiento y enriquecimiento de mensajes, complementado por funcionalidades de seguridad, monitorización y gestión [18]. El uso de un ESB es clave para el desarrollo de una arquitectura orientada a servicios y dirigida por eventos (ED-SOA), también conocida como SOA 2.0. Este tipo de arquitectura es una evolución de las arquitecturas orientadas a servicios tradicionales en la que la comunicación entre los usuarios y los servicios se realiza en base a eventos; esto es, los servicios son desencadenados por eventos y reaccionan a éstos, pudiendo ser fuente de nuevos eventos [17].

3. Arquitectura Propuesta

El objetivo principal de nuestra propuesta es que cuando se detecten niveles de calidad del aire potencialmente peligrosos para la salud del ciudadano, se le avise inmediatamente a través de un mensaje de alarma. El sistema de detección podría hacerle llegar este mensaje a través de diversos medios, pero dado el despunte de las tecnologías móviles, proponemos el desarrollo de una aplicación

específica que el ciudadano pueda instalar en su móvil, de modo que reciba un aviso en éste de manera inmediata cuando se detecte una calidad del aire indeseable para su salud. Esta información se proporcionará de manera genérica, pero si el usuario facilita información específica, a través de esta misma aplicación, sobre su edad, sus problemas de salud u otras peculiaridades que pudiesen verse afectados por la calidad del aire, se particularizará la información y se le enviarán las notificaciones que sean especialmente relevantes para él. Obviamente, independientemente de las notificaciones, el ciudadano podrá acceder en todo momento a la información actual sobre la calidad del aire. Tecnológicamente hablando, esto requiere definir e implementar un sistema que permita obtener y procesar información relativa a la calidad del aire proveniente de diversas fuentes en tiempo real; detectar posibles problemas derivados de la calidad del aire a través del procesado de dicha información e informar inmediatamente a los ciudadanos que pudiesen verse afectados por esta situación.

En este artículo partimos, en base a nuestra experiencia previa en el ámbito de la integración de IoT y CEP, de la arquitectura ya definida en nuestro trabajo previo [4]. Sin embargo, esta arquitectura ha sido refinada y ampliada en este artículo para lograr la consecución del objetivo principal previamente descrito. Como muestra la Fig. 1, la arquitectura integra los siguientes elementos:

- ESB que va a canalizar todas las comunicaciones: detección de eventos simples, enrutamiento de estos al motor CEP y a las bases de datos, detección y gestión de eventos complejos notificados por el motor CEP, invocación de los servicios web, almacenamiento de la información del registro del usuario y notificación a este cuando proceda.
- Plataforma física compuesta por una serie de sensores que nos van a proporcionar información sobre distintos componentes del aire en tiempo real en una localización determinada: ozono (O_3), partículas de polvo (PM), monóxido de carbono (CO), dióxido de carbono (CO_2), dióxido de sulfato (SO_2), dióxido de nitrógeno (NO_2), temperatura y humedad, entre otros.
- Información de distintas fuentes de información sobre calidad del aire a través de una plataforma IoT.
- Motor CEP donde se definen los patrones que se desean detectar en relación a la calidad del aire y donde se reciben los eventos redirigidos a través del ESB, que van a permitir detectar si se han dado dichos patrones. Base de datos (BD) NoSQL donde se almacenan los parámetros de calidad del aire que se van detectando, tanto de la plataforma física como de la plataforma IoT, así como los eventos complejos detectados. Estas BD no requieren estructuras de datos fijas y se caracterizan por proporcionar consistencia eventual, esto es, al finalizar una operación el sistema puede quedar inconsistente, pero se garantiza que tras un tiempo será consistente, premiando la disponibilidad frente a la consistencia de las operaciones. Además, son más fáciles de escalar horizontalmente que las BD SQL y muy eficientes para gestionar grandes cantidades de datos de forma distribuida.
- BD SQL donde se almacenan los datos del registro del usuario interesado en la calidad del aire, así como sus circunstancias personales (edad, enfermedades, etc.) si las ha proporcionado. Estas BD utilizan estructuras de tablas

fijas y proporcionan atomicidad, consistencia, aislamiento y persistencia; sin embargo, no son aptas para ser distribuidas a gran escala [21].
- Servicio web (SW) que permite consultar en todo momento los parámetros actuales de calidad del aire.
- SW que lanza las notificaciones al producirse situaciones potencialmente peligrosas para la salud del ciudadano en función de la calidad del aire.
- Aplicación móvil que puede enviar solicitudes de información sobre calidad del aire así como recibir notificaciones previo registro del usuario.

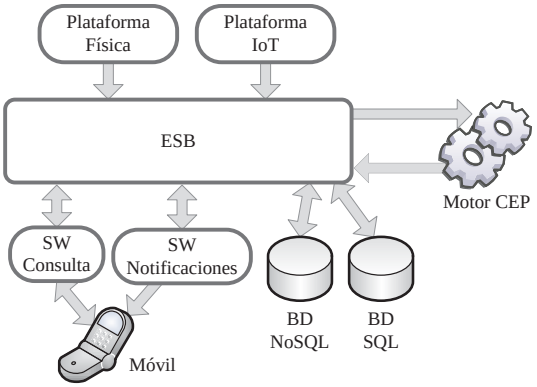

Fig. 1. Arquitectura para la prevención de riesgos derivados de la calidad del aire.

El funcionamiento de la arquitectura sería el siguiente. Por una parte el usuario debe descargarse e instalarse la aplicación en el móvil, desde la cual podrá consultar en todo momento la información de calidad del aire actualizada en tiempo real (esto se podría hacer también a través de un portal web). Al instalar la aplicación, o en cualquier momento posterior si lo desea, podrá registrarse en la aplicación para recibir notificaciones sobre situaciones de calidad del aire potencialmente peligrosas para su salud. Si en su registro no proporciona información personal, las notificaciones que recibirá serán generales, basadas en lo que la OMS considera parámetros de calidad del aire peligrosos para una persona adulta sin enfermedades relevantes relacionadas con el corazón o el sistema respiratorio. Si el usuario además hubiese proporcionado información personal (edad, enfermedades cardio-respiratorias, alergias...) entonces las notificaciones que recibirá serán particularizadas para su grupo de riesgo.

Por otra parte, el desarrollador de la aplicación habrá gestionado el formateado de los datos que llegan tanto desde la plataforma física como desde la de IoT, así como habrá insertado en el motor CEP los patrones que es necesario detectar. De este modo, llegarán constantemente, en tiempo real, los valores de calidad del aire y condiciones atmosféricas de ambas plataformas y serán procesados en el motor CEP detectando, en su caso, los eventos complejos de interés. Estos últimos, así como todos los parámetros de calidad del aire recibidos se

almacenarán en la BD NoSQL, tanto para poder ser consultados en el mismo momento por el usuario como para posteriores fines estadísticos. Estos valores podrán obtenerse a través del SW de consulta. Además, en el momento que se haya detectado un evento de interés se notificará automáticamente a los usuarios registrados a través del SW de notificaciones.

4. Prototipo y Caso de Estudio

4.1. Tecnologías del Prototipo

Para la implementación de la arquitectura propuesta se ha optado por el uso de: Mule como ESB, plataforma física compuesta de sensores de O_3, PM, CO, CO_2, SO_2, NO_2, temperatura y humedad, Xively como plataforma IoT, Esper como motor CEP, MongoDB como sistema de gestión de BD NoSQL, MySQL como sistema de gestión de BD SQL, Apache CXF para la implementación de los servicios web, y Android para la aplicación móvil. En este momento, el prototipo se encuentra en fase de desarrollo; en la Sección 5 detallamos qué partes se encuentran ya implementadas y en qué otras partes estamos trabajando.

4.2. Descripción del Caso de Estudio

Aún no tenemos en funcionamiento la plataforma física, de modo que el caso de estudio ilustrativo se ha implementado en base a la información recibida a través de la plataforma IoT. Igualmente, la aplicación móvil se encuentra en desarrollo por lo que las notificaciones, a efecto de comprobar que se realizan en tiempo real, se envían a la dirección de correo electrónico que haya proporcionado el usuario, sin necesidad de utilizar un SW como intermediario.

Así pues, en este caso de estudio, los productores de eventos son sensores localizados en distintas zonas geográficas, disponibles en la plataforma Xively: Popayán (Colombia) —disponible en `https://xively.com/feeds/2003665973`—, Detroit (EE.UU.) —disponible en `https://xively.com/feeds/23726`— y Boom (Bélgica) —disponible en `https://xively.com/feeds/101426`. Cada sensor dispone de un flujo de datos en donde se almacenará los datos (O_3, PM, CO, SO_2 y NO_2, temperatura y/o humedad) tomados por el sensor cada varios minutos. Los consumidores serán todos aquellos usuarios cuya dirección de correo electrónico haya sido almacenada en nuestra BD.

Atendiendo a los requisitos reales para detectar los niveles de calidad de aire, hemos definido 6 patrones de eventos (véase la Tabla 1). Las condiciones del patrón especifican cuáles son los valores entre los que debe estar comprendido el índice de calidad de aire para que sea detectado el patrón correspondiente. Téngase en cuenta que este índice de calidad se calcula de la siguiente forma: $índiceCalidadAire = máx\{O_3, PM, CO, SO_2, NO_2\}$.

4.3. Implementación del Caso de Estudio

Los pasos seguidos para implementar este caso de estudio son los siguientes:

Tabla 1. Patrones de eventos de calidad del aire.

Nombre del Patrón	Condiciones del Patrón
CalidadAireBuena	$0 \leq índiceCalidadAire \leq 50$
CalidadAireModerada	$51 \leq índiceCalidadAire \leq 100$
CalidadAireInsalubreParaGrupos	$101 \leq índiceCalidadAire \leq 150$
CalidadAireInsalubre	$151 \leq índiceCalidadAire \leq 200$
CalidadAireMuyInsalubre	$201 \leq índiceCalidadAire \leq 300$
CalidadAirePeligrosa	$301 \leq índiceCalidadAire \leq 500$

- **Integración de Xively con Mule**: Se ha creado una aplicación Mule que se encarga de recibir datos de los tres sensores conectados a Xively, previamente mencionados, cada varios minutos a través de un conector HTTP, transformar los datos recibidos en formato JSON a eventos *maps* de Java y, finalmente, enviarlos al motor Esper para su procesado, así como almacenarlos en una BD MongoDB. Además, cada evento será enriquecido, añadiéndose al *map* que lo representa la propiedad *indiceCalidadAire*, cuyo valor viene determinado por el cálculo de la fórmula previamente descrita.
- **Detección y notificación de los eventos complejos**: Los patrones de eventos descritos anteriormente han sido implementados utilizando el lenguaje EPL de Esper, el motor CEP seleccionado. Seguidamente, se muestra la implementación del patrón *CalidadAireBuen*a, a modo de ejemplo:

```
@Name('CalidadAireBuena')
insert into NivelesCalidadAire
   (tiempo, localizacion, indiceCalidadAire, nivel)
select buenNivel.tiempo, buenNivel.localizacion,
   buenNivel.indiceCalidadAire, 'bueno'
from pattern [every buenNivel = EventoT(
   indiceCalidadAire >= 0 and indiceCalidadAire <= 50)];
```

  Las condiciones que deben cumplirse para detectar el nivel de calidad de aire *bueno* se especifican en la cláusula *from pattern*. El operador *every* recorrerá cada uno de los eventos de tipo *EventoT* que lleguen al motor y se les asignarán el alias *buenNivel* cuando se encuentren eventos cuyo valor de *indiceCalidadAire* se encuentre comprendido entre 0 y 50. A continuación, se creará un evento complejo con los datos especificados mediante la cláusula *select* y se insertará en un nuevo flujo de eventos denominado *NivelesCalidadAire*. Los eventos complejos serán capturados por unos componentes específicos, denominados *listeners*, que se encargarán de enviarlos al ESB.
- **Integración de Mule con el servidor de correos**: Una vez lleguen al ESB los eventos complejos informando de los niveles de calidad de aire en cada zona geográfica, se comunicarán por correo electrónico a todos los que estén interesados en recibirlos. Para ello, se ha utilizado el protocolo SMTP. Además, en estos mensajes se podrán añadir recomendaciones a los ciudadanos según la calidad del aire que respiren como, por ejemplo, "reducir el tiempo en actividades al aire libre" en el caso de que la calidad de aire sea

insalubre. El envío de correo electrónico se ha implementado para la prueba del prototipo, en espera de la finalización del servicio de notificaciones.

5. Discusión

La agencia de protección medioambiental de los EE.UU. (EPA) ha definido un índice denominado *Air Quality Index* (AQI) [11] que informa diariamente sobre la calidad del aire de las zonas de los EE.UU., clasificada en niveles que oscilan desde valores "buenos" hasta "peligrosos" para la salud. Además, este índice especifica los problemas de salud que podrían aparecer en las próximas horas o días, en el caso de que una persona respire aire en malas condiciones. El caso de estudio ha permitido demostrar que CEP es una solución efectiva y alternativa para detectar en tiempo real (minuto a minuto) los distintos niveles de calidad del aire en distintas zonas geográficas a nivel mundial; en comparación con la herramienta EnviroFlash [12] de la EPA, que, como muchas otras, proporciona dicha información en intervalos de tiempo superiores y sólo en algunas regiones determinadas. Nótese que se ha puesto de ejemplo la EPA por ser una de las agencias más avanzadas en este sector a nivel mundial, si bien pueden encontrarse ejemplos más cercanos —en la Comunidad de Madrid, la Junta de Andalucía o Asturias, entre otros— pero mucho menos desarrollados. Por ejemplo, la aplicación Respira Xixon [2] no funciona en las versiones actuales de Android. El estado actual de desarrollo del prototipo propuesto es el siguiente:

- Se encuentran ya desarrolladas la integración del ESB con el motor CEP, así como con las plataformas IoT y BD NoSQL. El prototipo cuenta ya con un SW que permite consultar los parámetros de calidad del aire almacenados actualmente en la BD.
- En fase de desarrollo se encuentra la definición de los patrones que son relevantes para la población en general y para los grupos de riesgo en particular; si bien ya hemos realizado un estudio específico del dominio, nos queda pendiente la definición de los patrones en EPL. En fase de desarrollo también está la aplicación móvil.
- Queda pendiente de desarrollo la integración de la plataforma física, que se ha retrasado por problemas de *stock* y la implementación de las notificaciones particularizadas en función de la situación personal del usuario.

6. Trabajos Relacionados

En varios trabajos se ha propuesto la integración de IoT y CEP en SOA 2.0. Da et al. [9] proponen una arquitectura SOA que soporta servicios personalizados centrados en el usuario altamente escalables para IoT, haciendo uso de un ESB y el motor de reglas Drools [13]. En nuestra arquitectura propuesta utilizamos el motor CEP Esper, en lugar de un motor de reglas, ya que como afirman Chandy y Schulte [7] son más rápidos y eficientes.

Por otro lado, Walczak et al. [22] proponen un enfoque para comunicación máquina a máquina y procesamiento de datos en aplicaciones del Internet del futuro. En este caso, también se diferencia de nuestra propuesta en que utilizan el motor de reglas Drools. Además, tampoco usan un ESB para integrar IoT y CEP, mientras que en nuestra arquitectura hacemos uso de Mule, uno de los ESB de código abierto más populares en la actualidad, facilitando la integración de la arquitectura con otras aplicaciones y otras arquitecturas ya existentes.

Existen otros trabajos que integran IoT y SOA, pero no usan CEP para procesar la información. Por ejemplo, Sleman y Moeller [20] describen un sistema operativo distribuido basado en SOA que gestiona datos y eventos utilizando mensajes en formato XML y JSON. En su arquitectura se utiliza una cola de mensajes para distribuir y manipular dichos mensajes. Nuestra arquitectura además de CEP, utiliza un ESB que es más potente que una cola de mensajes y proporciona una mayor funcionalidad.

7. Conclusiones y Trabajo Futuro

En este artículo hemos propuesto una arquitectura que facilita la consulta y notificación al ciudadano en tiempo real de parámetros de calidad del aire potencialmente peligrosos para su salud. Además, se ha desarrollado un prototipo inicial capaz de gestionar, analizar y correlacionar una ingente cantidad de datos relativa a los distintos niveles de calidad de aire que se respiran en distintas zonas distribuidas geográficamente a nivel mundial con el objetivo de detectar en tiempo real situaciones críticas relacionadas con la calidad del aire y notificarlas al correo electrónico del usuario. En concreto, se han utilizado tres sensores localizados en Colombia, EE.UU. y Bélgica. Se ha comprobado que, en efecto, la arquitectura propuesta es capaz de detectar y notificar niveles de calidad de aire minuto a minuto, mientras que las herramientas existentes en la actualidad informan en intervalos de tiempo superiores.

Como trabajo futuro finalizaremos la implementación del prototipo. Adicionalmente, queremos integrar también en la aplicación un análisis estadístico de los datos registrados, así como la visualización gráfica de datos en tiempo real. Finalmente, propondremos el uso libre de nuestra plataforma a diversas empresas relacionadas con la meteorología con las que actualmente estamos en contacto, así como a las administraciones públicas a las que pudiese suscitar interés.

Agradecimientos. Este trabajo está financiado por el Ministerio de Ciencia e Innovación (proyecto TIN2011-27242).

Referencias

1. Air Quality in Europe. `http://www.airqualitynow.eu/` (2014)
2. Respira Xixon. `http://respiraxixon.com` (2014)
3. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A Survey. Computer Networks 54(15), 2787–2805 (octubre 2010)

4. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: An approach of early disease detection using CEP and SOA. In: Proceedings of The Third International Conferences on Advanced Service Computing. pp. 143–148. Rome, Italy (septiembre 2011)
5. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: Integración del Internet de las Cosas y las Arquitecturas Orientadas a Servicios: un Caso de Estudio en el Ámbito de la Domótica. In: Actas de las IX Jornadas de Ciencia e Ingeniería de Servicios. pp. 35–49. Madrid, Spain (septiembre 2013)
6. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: A model-driven approach for facilitating user-friendly design of complex event patterns. Expert Systems with Applications 41(2), 445–456 (febrero 2014)
7. Chandy, K.M., Schulte, W.R.: Event Processing: Designing IT Systems for Agile Companies. McGraw-Hill, USA (2010)
8. Comunidad de Madrid: Calidad del aire. `http://www.mambiente.munimadrid.es/opencms/opencms/calaire` (2014)
9. Da, Z., Bo, C., Yang, Z., Junliang, C.: Future Service Provision: Towards a Flexible Hybrid Service Supporting Platform. In: Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific. pp. 226 –233 (diciembre 2010)
10. Davis, J.: Open Source SOA. Manning Publications (mayo 2009)
11. EPA: Air quality index (AQI) - a guide to air quality and your health. Tech. Rep. EPA-456/F-09-002, U.S. Environmental Protection Agency, Research Triangle Park, NC (agosto 2009), `http://www.epa.gov/airnow/aqi_brochure_08-09.pdf`
12. EPA: EnviroFlash. `http://www.enviroflash.info/` (2014)
13. JBoss: Drools Fusion. `http://www.jboss.org/drools/drools-fusion.html` (2013)
14. Junta de Andalucía: Calidad del aire. `http://www.juntadeandalucia.es/temas/medio-ambiente/emisiones/calidad.html`
15. LogMeIn: Xively - Public Cloud for IoT. `https://xively.com/` (2013)
16. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Estados Unidos (2001)
17. Luckham, D.: Event Processing for Business: Organizing the Real-Time Enterprise. Wiley, New Jersey, USA (2012)
18. Rademakers, T., Dirksen, J.: Open-Source ESBs in Action. Manning, Greenwich (2009)
19. Rouil, L., Honoré, C., et al.: Prev'air: An Operational Forecasting and Mapping System for Air Quality in Europe. Bulletin of the American Meteorological Society 90(1), 73–83 (enero 2009)
20. Sleman, A., Moeller, R.: SOA distributed operating system for managing embedded devices in home and building automation. IEEE Transactions on Consumer Electronics 57(2), 945–952 (2011)
21. van der Veen, J., van der Waaij, B., Meijer, R.: Sensor data storage performance: SQL or NoSQL, physical or virtual. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD). pp. 431–438 (junio 2012)
22. Walczak, D., Wrzos, M., Radziuk, A., Lewandowski, B., Mazurek, C.: Machine-to-Machine Communication and Data Processing Approach in Future Internet applications. In: The 8th International Symposium on Communication Systems, Networks Digital Signal Processing. pp. 1–5 (julio 2012)
23. WHO: Review of evidence on health aspects of air pollution - REVIHAAP project (2013), `http://www.euro.who.int/__data/assets/pdf_file/0020/182432/e96762-final.pdf`

PERSEO: Identificando servicios en sistemas heredados mediante un enfoque ADM

Ignacio García-Rodríguez de Guzmán[1], Ricardo Pérez-Castillo[2], Mario Piattini[1]

Institute of Information Technologies and Systems[1]
University of Castilla-La Mancha
Camino de Moledores s/n, 13071, Ciudad Real
{Ignacio.GRodriguez, Mario.Piattini}@uclm.es
Itestra GmbH[2]
Capitán Haya 1, Planta 15, 28020 Madrid
perez@itestra.com
+49 177 277 4558

Resumen. En la actualidad, las empresas se encuentran con el hecho de que sus sistemas de información empiezan a encontrarse obsoletos y sin apenas capacidad de maniobra para afrontar los cambios tanto tecnológicos como negocio que pueden surgir (y surgirán). El principal problema de esta obsolescencia es la cantidad de conocimiento embebido en el portafolio de sistemas de las empresas. Esto hace que la opción de desechar los sistemas actuales y sustituirlos por otros nuevos sea una opción que no resulta viable. La arquitectura orientada a servicios, también conocida como SOA, puede verse como otra fase dentro de la evolución del software, y que permite dotar a la infraestructura software de las empresas de esa flexibilidad de que en estos momentos adolece. Por ello, es posible que ésta sea la opción más adecuada ante la disyuntiva que se presenta con los sistemas heredados, permitiendo que los mismos evolucionen hacia este paradigma tecnológico. Este trabajo presenta una versión inicial de un entorno para la generación de especificaciones en SoaML a partir de sistemas heredados, facilitando así la migración de estos sistemas hacia el paradigma SOA.

Palabras clave. Architecture-Driven Modernization, service elicitation, legacy system, service migration, SoaML, KDM.

1 Introducción

La arquitectura orientada a servicios, también conocida como SOA, puede verse como otra fase dentro de la evolución del software. Sin embargo esta evolución supone un cambio muy significativo en cómo las empresas deben implementar y suministrar sus funcionalidades a los usuarios [1].

En la actualidad, las empresas se encuentran con el hecho de que sus sistemas de información se encuentran obsoletos y sin apenas capacidad de maniobra para afrontar los cambios tanto tecnológicos como de negocio que pueden surgir y surgirán. El principal problema de esta obsolescencia es la cantidad de conocimiento embebido

(procesos y reglas de negocio) en el portafolio de sistemas de las empresas. Esto hace que la opción de desechar los sistemas actuales y sustituirlos por otros nuevos sea una opción que no resulta viable.

SOA, como paradigma tecnológico, permite dotar a la infraestructura software de las empresas de esa flexibilidad de que en estos momentos adolece. Es por esto que se hace imprescindible hallar una solución de compromiso entre la adopción de los nuevos paradigmas tecnológicos y la conservación de los sistemas actuales, que embeben el conocimiento de negocio de las organizaciones.

Probablemente, la opción más adecuada ante esta disyuntiva es permitir que los sistemas existentes evolucionen desde sus estados actuales hacia los servicios, para lo cual, es necesario aplicar reingeniería de forma que se puedan obtener nuevos sistemas orientados a servicios y que conserven la funcionalidad, las reglas y los procesos de negocio actuales (esto último cobra aun más importancia en el contexto de los servicios).

Por este motivo, en este artículo los autores presentan PERSEO, una herramienta para la migración de sistemas heredados hacia servicios. PERSEO es una herramienta basada en estándares de OMG[1] como KDM [2], QVT [3] y SoaML [4]. PERSEO se encuentra en una versión beta, sin embargo, ya permite abordar la ingeniería inversa de sistemas para su representación mediante modelos PSM y PIM [5] y una identificación preliminar de servicios para su especificación mediante SoaML. Además, el desarrollo de PERSEO se ha estructurado de forma que se puedan admitir distintas estrategias de identificación de servicios a partir de sistemas heredados según la naturaleza del sistema heredado o los requisitos del proyecto de migración.

El resto del artículo se estructura de la siguiente manera: la sección 2 presenta una visión general del estado del arte sobre migración de servicios; la sección 3 describe el proceso de migración ilustrando los niveles de abstracción que se contemplan a la hora de migrar un sistema a servicios y las transformaciones involucradas; la sección 4 muestra la herramienta PERSEO; por último, en la sección 5 se da una visión general del artículo en forma de conclusiones y las tareas que restan acometer en la evolución inmediata del desarrollo de PERSEO.

2 Estado del arte

En esta sección se estudiarán, de forma resumida, algunas de las propuestas con más impacto en el ámbito de la migración de los LIS (*Legacy Information System*) a SOA

Uno de los primeros aspectos que deben tenerse en cuenta (no sólo en la migración a SOA, sino en cualquier proceso de reingeniería) es *qué se espera del sistema destino.* Esta fase puede entenderse como *el estudio de las características funcionales del sistema destino* [6-8]. En este sentido, el método SOAMIG [7] se centra en la importancia del diseño de los servicios como resultado de la ingeniería directa de los servicios identificados en un LIS durante la fase de ingeniería inversa. Por otro lado, el método SMART [8] realiza, en primer lugar, el diseño de los servicios del sistema

[1] *Object Management Group - http://www.omg.org/*

destino en base a las funcionalidades potenciales identificadas en el sistema origen, y por otro lado, realiza una verificación de dichos servicios con los *stakeholders*.

Casi tan importante como realizar la evolución de un LIS a SOA, es determinar la viabilidad de esta migración, o lo que es lo mismo, determinar si se hace o no. Las prácticas más conocidas para determinar la viabilidad de la evolución se resumen en la propuesta de análisis coste-beneficio para proyectos de reingeniería de [9]. Esta propuesta ha sido muy empleada dentro de las técnicas de evolución LIS a SOA [10-12]. El método SMART emplea la técnica *options analysis for re-engineering* (OAR) [13] para determinar la viabilidad de la migración. En [13], se presenta una herramienta de soporte para la toma de decisiones en la elección de la estrategia de modernización de LIS a SOA desarrollada a partir del método SMART y el *framework* de toma de decisiones [14].

Para resolver el problema de la identificación de servicios en el LIS existen dos grupos de técnicas importantes: (i) modelado de los requisitos de negocio (top-down), y (ii) código heredado hacia funcionalidades de negocio (botom-up). En el primer enfoque, los procesos de negocio más importantes se diseñan a partir de la información identificada en la fase de comprensión del sistema heredado. Este proceso de negocio es dividido en actividades hasta que estas son directamente *mapeables* a funcionalidades identificables en el LIS. Por otro lado, cuando se parte del código heredado para identificar las funcionalidades de negocio, se emplea el código fuente como punto de inicio para el descubrimiento de la información de negocio [15-21].

Para la integración de un LIS en un contexto SOA pueden seguirse (principalmente) dos enfoques: integración de sistemas heredados o migración de sistemas heredados. En la integración de sistemas heredados, el código fuente de los sistemas no sufre grandes cambios (uso de técnicas como los *wrappers* y los *middlewares*). La integración de sistemas heredados se considera una estrategia rápida, con menos riesgos, más económica y sencilla, sin embargo el sistema sigue siendo tal y como es, sin ser modificado. En [22] [12] [23] pueden encontrarse algunas propuesta en este sentido.

Por otro lado, en el enfoque de migración de sistemas heredados (*una restructuración y modificación interna del sistema heredado en un nuevo sistema*), el código fuente del LIS sí que es sometido a transformaciones. Esta técnica es considerada como la más costosa, aunque puede efectuarse paulatinamente y a lo largo del tiempo, obteniendo como resultado una nueva versión del sistema sin los problemas que suele arrastrar un LIS por efecto del envejecimiento[24]. Existen varias técnicas para la migración desde LIS a SOA, como pueden ser *program slicing* [10, 25-27], transformación de modelos [6, 28, 29].

3 Descripción del proceso de migración hacia servicios

PERSEO es una herramienta que permite identificar servicios a partir de sistemas heredados. PERSEO se centra en la fase de ingeniería inversa del modelo de reingeniería en herradura y se basa en KDM, el cual permite realizar representaciones conceptuales abstractas de las diferentes vistas de la arquitectura de los LIS. Para ello, PERSEO estructura la información que extrae del sistema heredado en cuatro niveles

de abstracción. El paso de un nivel a otro se realiza mediante transformaciones desarrolladas en QVT [3] y de forma programática. La Fig. 1 representa gráficamente la arquitectura de PERSEO. A continuación, se definen los niveles de abstracción que considera PERSEO:

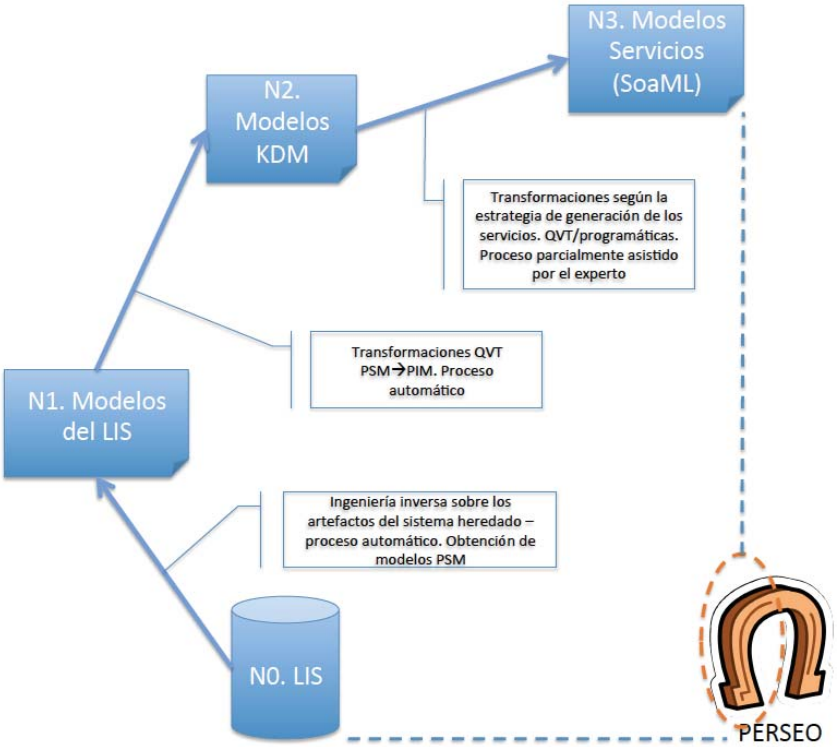

Fig. 1. Vista general de PERSEO

- Nivel 0 (N0). Este nivel representa al LIS de origen en el mundo real, del cual se pretende descubrir los servicios subyacentes.
- Nivel 1 (N1). Este nivel agrupa un conjunto de modelos que representan las diferentes vistas o aspectos de la arquitectura del LIS, es decir los diferentes activos software como código fuente, bases de datos, componentes, etc. La transformación N0-a-N1 obtiene modelos PSM desde cada activo software heredado. Para ello se recurre a la extracción de información del LIS por medio de técnicas de ingeniería inversa clásicas como análisis estático/dinámico, *slicing*, etc [30]. Los diferentes modelos PSM se construyen de acuerdo a metamodelos específicos según la tecnología del sistema origen o el artefacto del mismo que se esté procesando.
- Nivel 2 (N2). Este nivel se enfoca en la representación integrada de todos los activos software de un LIS de acuerdo al metamodelo de KDM. Durante la transformación N1-a-N2 los modelos específicos se transforman hacia un único modelo PIM basado en KDM. Estas transformaciones se formalizan mediante QVT. Dentro de este mismo nivel pueden establecerse transformaciones entre diferentes capas de la arquitectura KDM. Por ejemplo transformaciones entre la capa de ele-

mentos de programa o recursos que representan conocimiento explicito del LIS y la capa de abstracción que representa conocimiento implícito. De esta manera se reduce progresivamente la brecha conceptual entre el LIS y los servicios.

- Nivel 3 (N3). El último nivel representa los modelos de servicios que son inferidos a partir de los modelos KDM. El metamodelo que nos permite representar la información en este nivel es *SoaML* [4], un estándar de OMG para la especificación de sistemas basados en servicios. Cabe destacar, tal y como se ha mostrado en la sección 2, que existen distintas estrategias a la hora de migar un sistema heredado hacia SOA. Por esta razón, el desarrollo de la transformación N2-a-N3 no se plantea como una trasformación única. En la actualidad se está planteando el desarrollo de distintas transformaciones que permitan aplicar, según el contexto y las necesidades, una estrategia u otra para obtener un conjunto distinto de servicios.

La Tabla 1 resume cada una de las tres transformaciones de PERSEO. El objetivo final es completar el camino N0→N1→N2→N3 para generar los servicios necesarios que permitan exponer la funcionalidad más importante del sistema heredado.

Tabla 1. Transformaciones entre los niveles de PERSEO

Transformación		Artefactos	Metamodelos	Técnica de transformación
N0-a-N1	in	Activos software del LIS: código fuente, bases de datos, interfaces de usuario, etc.	-	Técnicas de ingeniería inversa: análisis estático y dinámico, *program slicing*, etc.
	out	Modelos PSM por cada activo software	Metamodelos específicos: MM_{JAVA}, MM_{SQL}, MM_{C++}, etc.	
N1-a-N2	in			Transformaciones entre modelos establecidas mediante *QVT Relations*.
	out	Modelos KDM tipo PIM	Metamodelo de KDM	
N2-a-N3	in			Transformaciones implementando estrategias de migración. Desarrolladas en QVT y de forma programática Intervención del experto para ayudar en la configuración de los servicios.
	out	Modelo de Servicios	Metamodelo de SoaML	

4 PERSEO

PERSEO se presenta como un *plug-in* que se integra en la plataforma de desarrollos Eclipse (ver Fig. 2 (izda)). PERSEO, que se muestra como una opción de menú en la que se implementan las diferentes etapas del proceso.

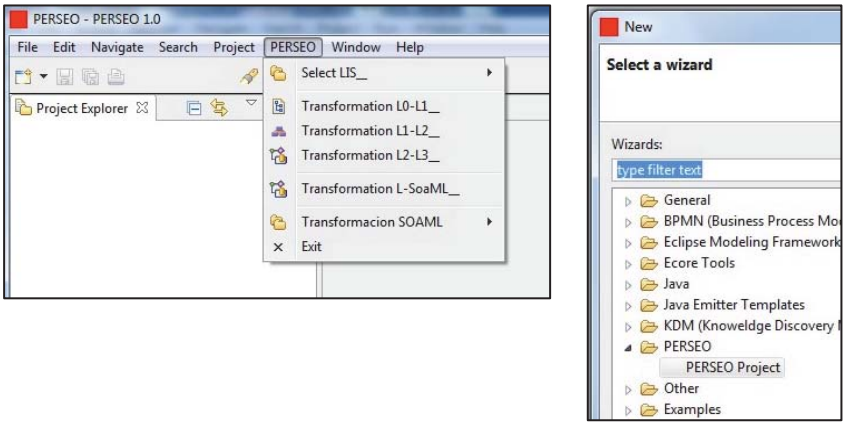

Fig. 2. Menú principal del *plug-in* PERSEO (izda) y creación de un nuevo proyecto PERSEO (dcha)

Para comenzar la identificación de servicios en un sistema heredado, se creará un nuevo proyecto del tipo "PERSEO Project" (ver Fig. 2 (dcha)), con lo que se dispondrá de un proyecto vacío con la arquitectura que se ha definido anteriormente. A continuación, se realizan los siguientes pasos previos a la identificación de los servicios:

1. Se carga el sistema heredado en el nivel L0. Todo el código fuente del sistema origen se ubica en este directorio para su procesamiento (ver Fig. 3(a)).
2. Se obtienen los modelos PSM del sistema heredado en el nivel L1 (ver Fig. 3(b)). Se ejecuta la transformación L0-a-L1 (ver Fig. 2 (dcha)).
3. Se obtienen los modelos KDM en el directorio L2 (ver Fig. 3(c)) mediante la transformación L1-a-L2 (ver Fig. 2 (dcha)).

La versión actual de PERSEO soporta la generación de especificaciones *SoaML* basada en el *wrapping* de funcionalidades del sistema identificadas a partir de los modelos KDM. La Fig. 4 muestra cómo, a partir de los modelos KDM, se pueden elegir qué funcionalidades serán presentadas como servicios y cuya información se empleará en la generación de los modelos *SoaML*.

Como se indicaba, la versión actual genera especificaciones básicas de *SoaML* sin embargo (y tal y como se exponía en la sección anterior) la transformación para pasar de los niveles N2 a N3 (de KDM a *SoaML*) no es una única, sino que dependerá de la estrategia de migración que se decida aplicar según el contexto del proyecto. Aunque la versión actual de la transformación es una versión "exploratoria", ya está en marcha el diseño de las implementaciones de la misma para otras estrategias de identificación de los servicios y su consecuente especificación.

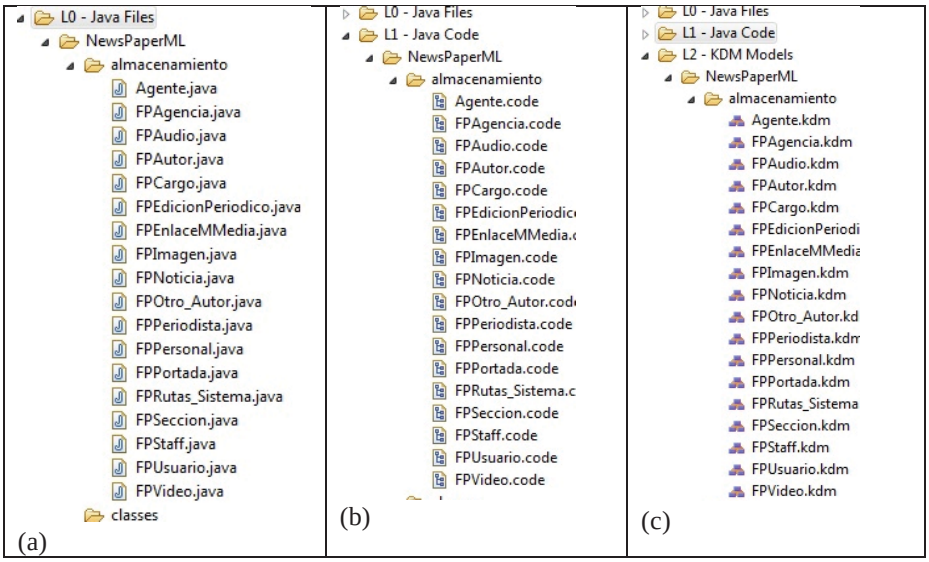

Fig. 3. Aplicación de las sucesivas transformaciones a partir de código (a) para obtener los modelos PSM (b) y PIM (c)

5 Conclusiones

La migración de sistemas heredados hacia servicios ha dejado de ser una promesa de futuro para convertirse en una realidad para las organizaciones actuales. La integración de sistemas y la apertura a nuevas oportunidades de negocio hacen que los Sistemas de Información deban situarse en la nube en forma de servicios, y para esto hay dos opciones (*grosso modo*): (i) los sistemas se desarrollan desde cero con esta orientación; o (ii) los sistemas existentes (la inmensa mayoría de los casos) que soportan la operativa de las empresas deben ser migrados o adaptados a SOA.

Como demuestra el estado del arte, existen múltiples situaciones y circunstancias a la hora de migrar un sistema heredado hacia un nuevo paradigma, y en función de las circunstancias y el contexto, unas opciones de migración serán más adecuadas que otras. Por este motivo, en este artículo se presenta PERSEO, una propuesta de herramienta que pretende ofrecer soporte al proceso de migración de sistemas heredados hacia servicios. La versión actual de PERSEO aborda la generación de especificaciones *SoaML* de forma simple, sin embargo, ya se está trabajando en el desarrollo de otras opciones y estrategias (o patrones) para la generación de servicios.

Principalmente, el trabajo futuro de PERSEO se mueve en dos direcciones: (i) la implementación de nuevas transformaciones para la generación de servicios en distintos modos, y (ii) el desarrollo de un módulo para la generación de código que permita obtener los servicios funcionales listos para desplegar en la nube, ya que la versión actual se centra en la identificación de servicios y su especificación.

Otro de los aspectos que se deberán considerar en PERSEO como parte y complemento al proceso de migración hacia servicios son los procesos de negocio. Tal y

como se puede ver en [6, 7] los procesos de negocio modelados en BPMN son también un complemento necesario para llevar a cabo una correcta migración.

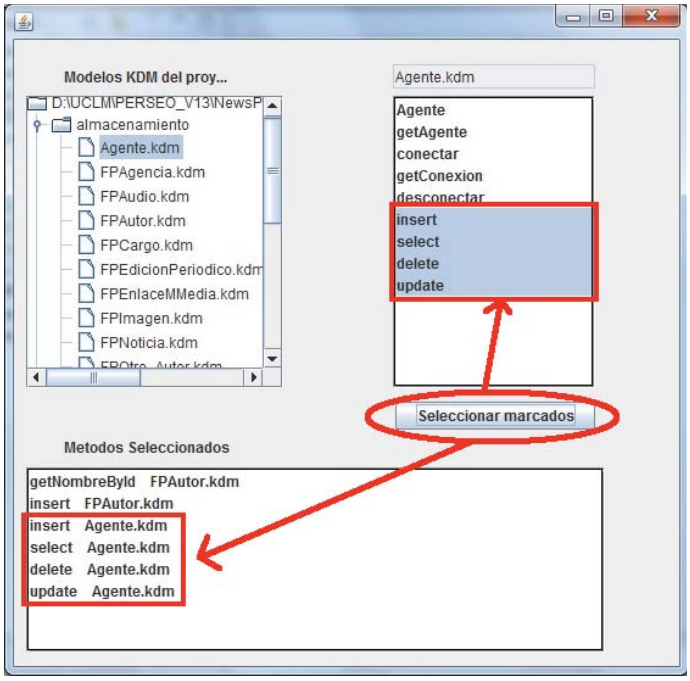

Fig. 4. Panel de configuración de los *MethodUnit* que se migrarán como servicios

6 Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto PISCIS: Paradigmas de Innovación, Social y Colaborativo aplicados a Ingeniería del Software (CDTI, FEDER - INNTERCONECTA – ANDALUCÍA, ITC- 20131007).

7 Referencias

1. Hutchinson, J., et al., *Migrating to SOAs by Way of Hybrid Systems.* T Professional. IEEE Computer Society, 2008. **10**(1): p. 34-42.
2. ISO/IEC, *ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization).* 2009, ISO/IEC. p. 302.
3. OMG, *QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification.* http://www.omg.org/spec/QVT/1.0/PDF, 2008, OMG.
4. OMG, *Service Oriented Architecture Modeling Language (SoaML),* 2012, Object Managent Group.
5. OMG, *MDA. Model Driven Architecture Guide Version 1.0.1*, 2003.

6. Fuhr, A., et al., *Model-driven software migration into service oriented architectures.* Computer Science - Research and Development, 2011. **28**(1): p. 65-84.
7. Zillmann, C., et al. *The SOAMIG Process Model in Industrial Applications*. in *In proceedings of the 15th European Conference on Software Maintenance and Reengineering*. 2011.
8. Lewis, G., E. Morris, and D. Smith. *Analyzing the reuse potential of migrating legacy components to a service-oriented architecture*. in *In Proceedings of the 10th European Conference on Software Maintenance and Reengineering, 2006. CSMR 2006*. 2006.
9. Sneed, H.M., *Planning the reengineering of legacy systems.* IEEE Software, 1995. **12**(1): p. 24-34.
10. Khadka, R., et al. *A Method Engineering based Legacy to SOA Migration Method*. in *In Proceedings of the the 27th IEEE International Conference on Software Maintenance (ICSM'11)*. 2011.
11. Sneed, H.M. *COB2WEB a toolset for migrating to web services*. in *In Proceedings of the 10th International Symposium on Web Site Evolution (WSE'08)*. 2008.
12. Sneed, H.M., I.J.S. Tools, and Technol. Transf., 441-451., *A pilot project for migrating COBOL code to web services.* International Journal on Software Tools for Technology Transfer, 2009. **11**(6): p. 441-451.
13. Smith, H., et al., *The Emergence of Business Process Management*, 2002, CSC´s Research Services Report.
14. Erradi, A., S. Anand, and N. Kulkarni. *Evaluation of Strategies for Integrating Legacy Applications as Services in a Service Oriented Architecture*. in *In Proceedings of the IEEE International Conference on Services Computing (SCC 06)*. 2006. Chicago, EEUU: IEEE Computer Society.
15. Aversano, L., L. Cerulo, and C. Palumbo. *Mining candidate web services from legacy code*. in *In Proceedings of the 10th International Symposium on Web Site Evolution, (WSE'08)*. 2008. Beijing: IEEE Computer Society.
16. Zhang, Q., R. Chen, and Y. Zou. *Reengineering User Interfaces of E-Commerce Applications Using Business Processes*. in *Proceedings of the 22st IEEE International Conference on Software Maintenance (ICSM'06)*. 2006. Philadelphia, Pennsylvania: IEEE Computer Society.
17. Zhang, Z. and H. Yang. *Incubating Services in Legacy Systems for Architectural Migration*. in *In Proceedings of the 11th Asia-Pacific Software Engineering Conference (ASPEC'04*. 2004.
18. Marchetto, A. and F. Ricca, *From objects to services: toward a stepwise migration approach for Java applications.* International Journal on Software Tools for Technology Transfer (STTT), 2009. **11**(6): p. 427-440.
19. García-Rodríguez de Guzmán, I., M. Polo, and M. Piattini. *An ADM Approach to Reengineer Relational Databases Towards Web Services*. in *14th Working Conference on Reverse Engineering (WCRE 2007)*. 2007. Vancouver, British Columbia, Canada: IEEE Computer Society.

20. Jiang, Y. and E. Stroulia. *Towards Reengineering Web Sites to Web-services Providers*. in *Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04)*. 2004. Tampere, Finland: IEEE Computer Society.
21. Zhang, Z., et al., *A service composition approach based on sequence mining for migrating e-learning legacy system to SOA.* International Journal of Automatic Computing, 2010. **7**(4): p. 584-595.
22. Almonaies, A.A., J.R. Cordy, and T.R. Dean. *Legacy system evolution towards service-oriented architecture*. in *In Proceedings the Interrnational Workshop on SOA Migration and Evolution (SOAME 2010)*. 2010. Madrid (España).
23. Zhang, B., et al. *A Black-Box Strategy to Migrate GUI-Based Legacy Systems to Web Services*. in *In Proceedings of the 2008 IEEE International Symposium on Service-Oriented System Engineering (SOSE '08)*. 2008. IEEE Computer Society.
24. Visaggio, G., *Ageing of a data-intensive legacy system: symptoms and remedies*. Journal of Software Maintenance and Evolution: Research and Practice, 2001. **13**: p. 281-308.
25. Bao, L., et al. *Extracting Reusable Services from Legacy Object-Oriented Systems*. in *In Proceedings of the 2010 IEEE International Conference on Software Maintenance (ICSM)*. 2010. Timisoara: IEEE Computer Society.
26. Chen, F., et al. *Service Identification via Ontology Mapping*. in *In Proceedings of the 33th IEEE International Computer Software and Applications Conference*. 2009. Seattle,Washington (EEUU): IEEE Computer.
27. Marchetto, A. and F. Ricca. *Transforming a java application in an equivalent web-services based application: toward a tool supported stepwise approach*. in *In Proceedings of the 10th International Symposium on Web Site Evolution (WSE'08)*. 2008. Beijing: IEEE Computer Society.
28. Chen, F., et al. *A Formal Model Driven Approach to Dependable Software Evolution*. in *In Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC '06)* 2006. Chicaco, EEUU: IEEE Computer Society.
29. Hoyer, P., et al. *A Model-Driven Development Approach for Service-Oriented Integration Scenarios.* in *In Proceedings of the 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns.* 2009. IEEE Computer Society.
30. Canfora, G. and M. Di Penta. *New Frontiers of Reverse Engineering*. in *2007 Future of Software Engineering*. 2007. IEEE Computer Society.

Embedding Widgets-as-a-Service into Dynamic GUI

Jesús Vallecillos, Javier Criado, Luis Iribarne, and Nicolás Padilla

Applied Computing Group, University of Almería, Spain
{jesus.vallecillos,javi.criado,luis.iribarne,npadilla}@ual.es

Abstract. The service-oriented computing offers an ideal development framework for carrying out business processes related to the dynamic management of component-based web user interfaces. This article proposes an architecture for specification, storage, management and visualization of web user interfaces built from widgets that follow the recommendation of the W3C. It describes a Widgets-as-a-Service (WaaS) approach for interface deployment and a three-level data model for the definition of components that take part in the architecture. In addition, it shows some particularities of the used technology and the implementation developed. To illustrate this proposal, an example of WaaS-based graphical interface developed for the Environmental Information Network of Andalusia (REDIAM) is shown.

Keywords: components, architectures, widgets, Wookie, WSDL, GUI

1 Introduction

The software services available on the web are increasingly elements that need to be changed, updated and adapted to the users' demand. Web interfaces do not get out of this necessity and also require the service they offer as an interface to be dynamic and adaptive to the user. With this aim, new projects and proposals have come up in the last few years to build customized web interfaces through the configuration of widgets that the user wants to visualize [6]. For these applications, the user has one or more graphical interfaces available that he/she can configure to create some kind of dashboards. These interfaces are built according to graphical components of high or medium granularity (that is, they are not simple buttons or text fields) that group together some functionalities related to each other and give rise to mashup applications based on widgets [16]. Some of the most interesting and currently supported projects are Netvibes (`http://www.netvibes.com`), MyYahoo (`https://my.yahoo.com`) or Ducksboard (`https://ducksboard.com`).

Within this context, we became interested in our research focused on dynamic management of component-based UIs. From our perspective, the representation of this type of interfaces can be "abstracted" in order to create a definition through fragments or pieces analogously to a *bottom-up* approach within the *Component-based Software Engineering* (CBSE). At subsequent steps, we can

manipulate this abstract representation to adapt it to the changes according to the user's preferences or any other changes derived or not from the interaction and then make again the graphical interface shown to the user. This mechanism allows us to work with simplified models of interfaces and later obtain new widgets to be shown as a service dynamically embedded into the GUI.

On the other hand, the components in most widgets-based GUI proposals are isolated from each other, that is, there is no relationship or communication between them. Those proposals that do not deal with isolated components have the difficulty of communication, especially in the web domain, as communication between some elements may cause some security problems that must be taken into account and solved at design time when programming widgets and application [8]. In this sense, our approach certainly allows relationships and communication between widgets and suggests an indirect communication mechanism by means of a service that mediates between their components.

To illustrate our proposal, Figure 1 shows one part of the widget-based web interface that is being developed for the Environmental Information Network of Andalusia (REDIAM) as a transfer of the research results within the framework of ENIA project, a project of excellence funded by "Junta de Andalucia" (ref. TIC-6114). This project deals with geographic information and users should have a GUI that enables them to access to the multiple services available. For this, each user has access to a graphical interface that allows him/her to manage different widgets that offer certain services such as loading layers of geographic information on a map from OGC services (`http://www.opengeospatial.org`) provided by REDIAM. We also consider these widgets as services to which users can access. A reduced prototype is available at `http://acg.ual.es/enia/jcis`.

The remainder of the article is structured as follows. Section 2 shows the structure of the WaaS model for UI, in which we analyze the various layers of the architecture developed and the data model to define the components. Section 3 shows the implementation options for its deployment. Section 4 revises some related work. Finally, Section 5 presents the conclusions and future work.

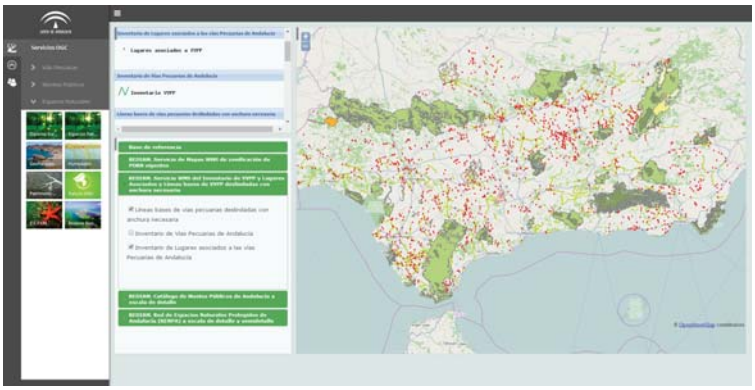

Fig. 1. Web application built from components

2 Widgets-as-a-Service (WaaS): Architecture description

In order to offer Widgets-as-a-Service (WaaS), we needed to develop an architectural system to support it. Furthermore, to isolate the proposal from the domain and with the aim of being able to manage component-based applications for different platforms, we built the system according to a three-layer architecture. Let us see in this section some features of the layers of this architecture as well as the used data model.

2.1 Architecture Layers

The client layer is the upper layer of the architecture. In our case, the client uses the services that the architecture deploys, coming from the "platform dependent" layer. In our example domain, the client is a web application, which means that this application has been developed under this technology and it must be accessed through a browser. Such application is built on the basis of components that, as it is web technology, have been implemented through widgets.

The widgets in which the application has been developed follow the recommendation of widget of the W3C. Moreover, the application is supported on the services that the architecture deploys to obtain the functionality it requires. By means of these services of the system architecture, the component-based web application can be initialized, receive support for the communication between components and give support to the components that form the application.

The servers dependent on the platform constitute the intermediate layer of the system architecture. This layer deals with providing the client with the required services and interacting with the independent part of the platform (the bottom level), thus getting some services from it and providing it with others. Regarding the first ones, this layer gets the code necessary to create the start-up web application and consults the path to be followed by the information to communicate the components with each other. We achieved this by using web services implemented in the independent part of the platform. On the other hand, this layer provides to the independent part of the platform with some information about the components it manages, such as the direction of the widgets instances which it will use to create the code defined by the web application. In these layers we can find not only the web client that carries out the functions of the user graphical interface with the system, but also those servers that have been specifically developed to use our proposal within the domain of component-based web user interfaces.

On the other hand, the components managed by our architecture are "black-box" components, in which their behavior is hidden and the component is represented by a specification template. This specification describes both the functional part and the extra-functional part (as well as some additional information) of components and this is the only way through which the system can make use of them. Therefore, the only components managed by our system are those defined and built according to this specification, which have also been registered for use. In order our component specification to be valid for

any type of components of third parties, we have extended the specification based on COTS (Commercial Off-The-Shelf) components [7]. For further details about the structure and content of the component specification used, please see `http://acg.ual.es/definitions/concreteComponent/component.xsd`.

Furthermore, we developed the architecture of our system by using a platform independent layer. This layer has a server that provides the system services that are valid for all platforms. For this, its functionalities are based only on the description of components and their relationships, regardless of the platform where they will be deployed. It is true that, at the deployment level, there are certain characteristics of each platform that should be taken into account. For instance, the initialization of a web user interface architecture may be different from any other component-based architecture implemented in Java; or the invocation of methods between components, as part of a communication task, might be different as well. However, there is a common area, an abstract view of such behaviors that can independently be extracted and implemented from the platform. As regards the above examples, we referred to "component architecture initialization" or "communication management", respectively.

Therefore, the services offered in this layer are the same for all platforms, hiding the distinction between different platforms and allows our proposal to be modular and scalable, being able to gradually add new functionalities for new platforms supported by the system. As this server has the core of the proposal for the management of component-based architectures, it has been named as *COScore* (Component-based architecture Operating Support).

2.2 Component Data Model

Although this article is based on dynamic deployment of widgets in web user interfaces, our ultimate goal is that the system can be able to manage component-based architectures for different platforms. Moreover, another objective is that the components managed by the system should not be just those predefined by the proposal developers but external developers and third-parties can also take part in the extension and update of repositories. For both reasons, we developed a three-level data model as shown in Figure 2 (described as follows).

The ECR **(External Component Repositories)** level corresponds to the external repositories of components, that is, the repositories of third-parties. These components are stored in their place of origin. This level is dependent on the platform because the components that it stores are specific for the mechanism they are assigned to. The ECR set is formed by each of the external repositories (ER) to be taken into account, that is, $ECR = \{ER_1, ER_2, ... , ER_n\}$.

The next level MCR **(Managed Component Repositories)** has a set of repositories of components managed by the system. This level stores the components that meet the structure and the construction guidelines established. It is also dependent on the platform since it is formed by modified versions of the ECR components so that they can be managed by our system and by a set of components developed for the system (MR_0). Therefore, $MCR = \{MR_0, MR_1, MR_2, ... , MR_n\}$, being MR_i $(i > 0)$ a subset of ER_i for which the

components have been embedded with the information and behavior required. At the same time, each MR repository is formed by a set of C components and a set of CI instances created and associated to such components. Each instance is created with a piece of information related to the user or client for whom the component was instantiated. This information contains, among other data, the instance identifier, the associated user and the component state. With this information, the system will be able to provide each user with the corresponding set of instances in the proper state.

Finally, the CS **(Component Specifications)** level stores component specifications. These specifications are referenced from the models that the component architectures describe so as to indicate which component should be chosen for its construction. When a developer registers one component in the system, he/she must register not only the component in the corresponding MR repository but its specification, too. That is the only way through which the system can take into account this component for future architecture construction and so that it can be embedded into a resulting component-based application. These specifications are managed by the platform independent server.

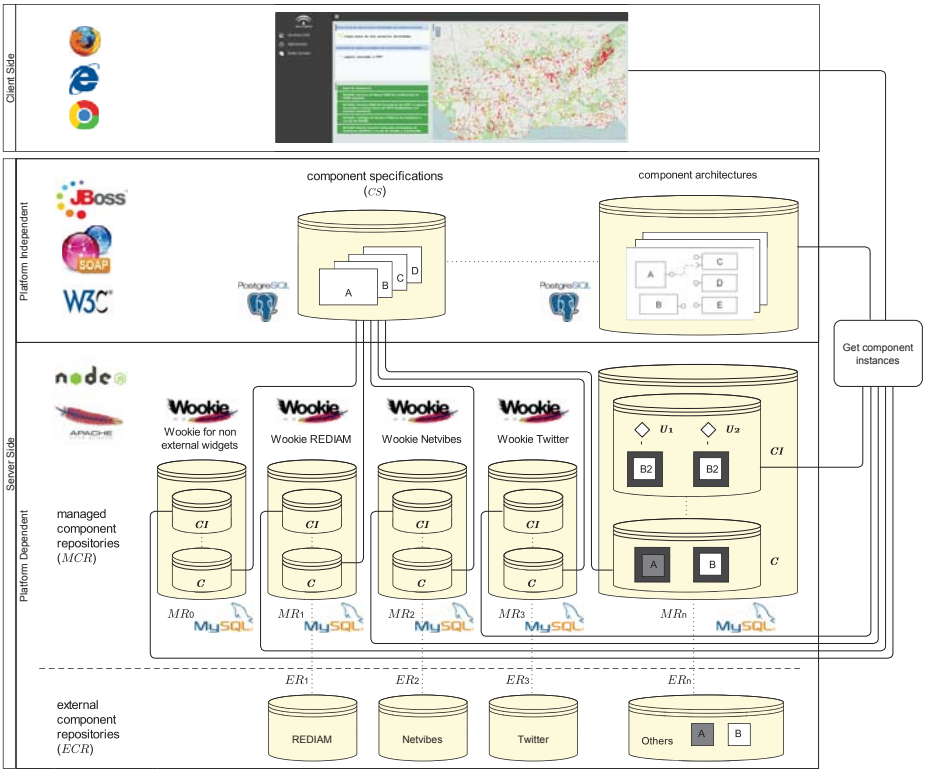

Fig. 2. Component Data Model for our widget-based user interfaces

Our proposal is oriented to the deployment of services in the cloud. Therefore, this data model seems to be suitable for the use of cloud technologies, being able to transfer each layer to different servers or even deploy each repository in different servers. In this way, the components, the instances and even the specifications will be considered from the point of view of "resource-as-a-service" (RaaS). In this scenario, the level that is dependent on the platform corresponds to the domain of web applications and consequently the repositories managed by the system (MCR) store widgets. These widgets either have been specifically developed for the system (MR_0) or built from external repositories (ECR). In the second case, two different situations may occur:

(a) that the external repositories store widgets, in which case we should create a *wrapper* upon the existing widget that can perform operations on it in order to be managed by our system:
(b) or that the external repositories store a service that is not a widget, in which case we need to create a widget that uses the service concerned.

Figure 2 shows how the system can manage external widgets created from services of REDIAM (`http://www.juntadeandalucia.es/medioambiente/site/rediam`), Netvibes or Twitter (`https://twitter.com/settings/widgets`). In this way, the system has all the elements available to be able to manage the widgets as a service (Widgets-as-a-Service) and dynamically build the web UIs.

3 WaaS-based Architecture deployment

Let's see some implementation details to be able to understand the structure of the architecture proposed. So, let us describe the technologies used in the chosen domain: component-based web user interfaces. We provided some information about the deployment from the client-side and from each of the servers, both dependent on the platform and the independent server. On the other hand, we also showed some code fragments to illustrate the behavior developed.

On the platform independent layer, a JBoss application server has been deployed (`http://www.jboss.org/jbossas`). Such application server offers a series of web services developed with JAX-WS [2]. These services are called from the layer that is dependent on the platform through SOAP messages [5] and offer the functionalities of the independent layer. All modules (capacities) of COScore are implemented via EJB [9] and internally managed through the lookup mechanisms (`lookup`) of this server and through different types of session (`stateful`, `stateless` and `singleton`) of *beans*.

On the platform dependent layer, we have deployed two servers: an Apache Wookie server (`http://wookie.apache.org`) and a Node.js server (`http://nodejs.org`). The first server allows deploying widgets based on the specification of the W3C (`http://www.w3.org/TR/widgets/`). The components reside in it for the case of web platform, where user graphical interfaces are built from widgets. This server provides an API (`http://wookie.apache.org/docs/api.html`) based on REST services [12] for the management of widgets (insertion,

elimination, modification, creation of instances, etc.). This server is deployed at `http://acg.ual.es/wookie` and an example of `Map` component for the user `user1` would be:

```
http://acg.ual.es/wookie/deploy/acg.ual.es/wookie/widgets/Map/
index.html?idkey=tuV6YYw9ztkjVRHkTC6NDz25D.sl.Y.eq.&proxy=http:
//acg.ual.es:80/wookie/proxy&st=
```

In this link, the clause `acg.ual.es/wookie/widgets/Map` is the component identifier and the code `tuV6YYw9ztkjVRHkTC6NDz25D.sl.Y.eq.` the user's instance identifier.

The second server is a JavaScript server that can be used as a link between the client and the platform independent server. As described, all communication from the client towards the lower layers goes through the Node.js server. The main reason for choosing this kind of server is because it can handle the received requests very efficiently and concurrently as it is event-oriented. Moreover, it allows us to send information from the server to the client side asynchronously, thus enabling the capacity for making changes in the interface, such as loading, removing, resizing or redistributing their components.

In order to manage communications with Node.js we use sockets. This requires to install the `socket.io` module (`http://socket.io`). We can see an example of this use in the code shown in Listing 1-(a). In the code, the Node.js server establishes an input port `initGUI` in its initialization (`connection`). This input port is invoked from the client web application to initialize the UI and, consequently, the web service of the COScore, in charge of getting the components, will be called. The result is sent through the output port `addComponent` to the web client. On the other hand, to invoke the SOAP-based web services, we need to install the `node-soap` module (`https://github.com/vpulim/node-soap`). In the code fragment shown in Listing 1-(b), we make use of this module to create a SOAP client that invokes the corresponding method of the web service and returns the result obtained.

To describe the client layer deployment, we are going to use as an example the implementation of the GUI initialization. At the client-side, it is necessary to include the code shown in Listing 1-(c) which is in charge of starting the connection between the web application and the *Node.js* server. Next, through the `initGUI` port, it sends such server a message with its user identifier to initiate the interface. On the other hand, it also establishes an input port named `addComponent` that will be called from the code described in Listing 1-(a) and will be in charge of adding the widgets to the web interface. As a result of the initialization, we can see a code fragment of the resulting widget-based application in Listing 1-(d). Finally, in order the client-side widgets can communicate with Node.js server, a connection to such server and a declaration of the input and output ports must be established. In Listing 1-(e), we can see an example of connection of the `Map` component as well as the declaration of an input port named as `loadLayer`. This implementation of ports is a special case for web platforms and it corresponds to the interfaces that the COScore server manages.

Listing 1. Code fragments of the architecture implementation

(a) Communication management from Node.js

```
io.sockets.on('connection', function (socket) {
  socket.on('initGUI', function(data, fn) {  ...
    callWS('http://acg.ual.es:8080/cos/COSWS?wsdl', 'initGUI',
    args_initGUI, function(ws_response) {
      if(ws_response != 'Error') {
        ws_response.forEach(function(value, index) {
         io.sockets.in(users[data.userID]).emit('addComponent', value); });
        fn(true);
      } else fn(false); });  });
});
```

(b) Web service invocation from Node.js

```
function callWS(wsurl, methodname, args, callback) {
  soap.createClient(wsurl, function(err, client) {
    if(client==null) { callback('Error'); }
    else client[methodname](args, function(err, result) { callback(result.return); });
  });
}
```

(c) GUI initialization from Client

```
var socket = io.connect('http://acg.ual.es:6969');
socket.on('connect', function(){
  socket.emit('initGUI', { userID: userID }, function(confirmation){
    if(!confirmation) alert('Error in GUI initialization'); });
});
socket.on('addComponent', function(data){ $('body').append(data); });
```

(d) Resulting web application

```
<html> ...
  <body>  ...
    <iframe type="text/html" src="http://acg.ual.es/wookie/deploy/acg.ual.es/wookie/widgets/
      Legend/index.html?idkey=u0MfpWWU3eT057fkAeVkb4GKzwY.eq.&proxy=http://acg.ual.es:80/
      wookie/proxy&st=&userID=user1&nodejsURL=acg.ual.es:6969"></iframe>
    <iframe type="text/html" src="http://acg.ual.es/wookie/deploy/acg.ual.es/wookie/widgets/
      Options/index.html?idkey=YRdICRqjAYRJ7z1tkWFV7Xl3HNg.eq.&proxy=http://acg.ual.es:80/
      wookie/proxy&st=&userID=user1&nodejsURL=acg.ual.es:6969"></iframe>
    <iframe type="text/html" src="http://acg.ual.es/wookie/deploy/acg.ual.es/wookie/widgets/
      Map/index.html?idkey=tuV6YYw9ztkjVRHkTC6NDz25D.sl.Y.eq.&proxy=http://acg.ual.es:80/
      wookie/proxy&st=&userID=user1&nodejsURL=acg.ual.es:6969"></iframe>
  </body>
</html>
```

(e) Widget connection and communication ports

```
var websocket = io.connect(getUrlVars()['nodejsURL']);
websocket.on('connect', function(){ websocket.emit('adduser', getUrlVars()['userID'],'Map', '');});
websocket.on('loadLayer', function(data){
   var newData = new Array();
   for (var i = 1; i < data.length; i++) newData[i-1] = data[i];
   newLayer(data[0], newData);
});
```

4 Related work

Next we are going to revise some works focused on the development of architecture systems that manage user interfaces built from components. We also present some works that analyze how to establish communication between components.

In [4] we can observe an architecture proposal where a deployment of web-based components is carried out. The authors developed an architecture to include dynamic applications through a web-based system. They studied the architecture through layers, one of which is in charge of managing the communication between the components and the system core. This form of communication is similar to the one explained in this work. Unlike this work, they did not focus on using the web standard of widgets of the W3C but rather on the use of Portlet [10] to build the applications.

We can find other works specialized in defining architectures for the management of widgets such as [3]. Here the authors proposed a new type of system for widget-based digital television platforms. They focused on widgets that are built with web technology rather than widgets that follow the web standard of the W3C. Unlike the means of communication described here, based on a JavaScript server, the authors used a mode of communication via AJAX to obtain the information stored in XML in a server. Another platform where we can use widget-based architectures is mobile applications. In [11] the authors defined an architecture made up of a widget component container that manages the interaction and a software platform that manages its dynamic life cycle. These widgets operated by the architecture are specific for the mobile platform which we are working with.

In [15] the authors focused on another functionality covered in this article, the communication between components. They analyzed the features of building mashup UIs by using widgets of the W3C and carried out a proposal to extend the widget model. The reason why they aimed to extend this model is to give support to a variety of patterns of communication between components. In contrast, our proposal developed a communication mechanism that did not need a revision of the component model. A similar work to the previous one is [14]. The communication between widgets is based on events through a defined API. For this, the authors implemented a system based on PHP and MySQL, where a container widget is in charge of managing the information flow by using JavaScript as technology. However, our proposal describes the communication messages as events by using a JavaScript server that manages that communication.

There exist other works like [13] which make adaptations of the code of the graphical user interface according to the user's preferences. The authors made the GUI more dynamic by developing it according to widget components. Unlike the widget components based on the web standard of the W3C used in this proposal, they implemented these components by making use of Java Swing. Moreover, they defined seven types of roles in order to improve the adaptation and reloading of widgets. Depending on the type of role, there are widgets considered as preferential to make their adaption.

5 Conclusions and future work

In this article we present a three-layer architecture for the specification, storage, management and deployment of component-based applications. Our aim is to abstract the definition and manipulation of such applications to describe behaviors that can be valid in multiple platforms. Within this architecture, the client layer deploys the applications or communicates with the rest of the system functionalities through services offered by the layer dependent on the platform. In turn, this second layer communicates with the platform independent layer, which offers services like application initialization and management of communication between components. On the other hand, we also describe the component data model developed within the architecture.

To validate this proposal, we have used the web domain as an application platform. In this way, we described the use of widgets as a service (Widgets-as-a-Service) to dynamically build component-based GUIs. We also described the communication management for such web components through a JavaScript server that plays the role of intermediary. As future work, we would like to increase the number of scenarios and validate, through practical examples, the proposed architecture in other platforms. We also aim to improve the user's experience when managing widget-based graphical user interfaces [1].

Acknowledgments

This work was funded by the Andalusian Regional Government (Spain) under Project P10-TIC-6114 and the Spanish Ministry of Education, Culture and Sport (MECD) under a FPU grant (AP2010-3259). This work was also supported by the CEiA3 and CEIMAR consortiums.

References

1. Cechanowicz, J., Gutwin, C.: Augmented Interactions: A Framework for Adding Expressive Power to GUI Widgets. INTERACT 2009, pp. 878–891. Springer (2009)
2. Chinnici, R., Hadley, M., Mordani, R.: The Java API for XML-Based Web Services (JAX-WS) 2.0. Specification JSR, 224 (2006)
3. Fan, K., Tang, S., Liu, Y., Zhang, S., Wang, Y., Xu, Z.: A System Architecture of Widget-Based Digital TV Interactive Platform. ICGEC, pp. 360–363. IEEE (2012)
4. Gmelch, O., Pernul, G.: A Generic Architecture for User-Centric Portlet Integration. 14th CEC, pp. 70–77. IEEE (2012)
5. Graham, S., et al.: Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI. SAMS publishing (2004)
6. Hoyer, V., Fischer, M.: Market overview of enterprise mashup tools. ICSOC 2008, pp. 708–721. Springer (2008)
7. Iribarne, L., Troya, J.M., Vallecillo, A.: A trading service for COTS components. The Computer Journal, 47(3), 342–357 (2004)
8. Jackson, C., Wang, H.J.: Subspace: secure cross-domain communication for web mashups. 16th WWW, pp. 611–620. ACM (2007)
9. Johnson, R.: J2EE development frameworks. Computer, 38(1), 107–110 (2005)
10. Law, E., Müller, D., Nguyen-Ngoc, A.: Differentiating and Defining Portlets and Widgets: A survey approach. MUPPLE'2009, pp. 123–131 (2009)
11. Pierre, D., Marc, D., Philippe, R.: Ubiquitous Widgets: Designing Interactions Architecture for Adaptive Mobile Applications. DCOSS'2013, pp. 331–336 (2013)
12. Richardson, L., Ruby, S.: RESTful web services. O'Reilly Media, Inc. (2008)
13. Shirogane, J., Iwata, H., Fukaya, K., Fukazawa, Y.: GUI Change Method according to Roles of Widgets and Change Patterns. IEICE Transactions on Information and Systems, 91(4), 907–920 (2008)
14. Sire, S., Paquier, M., Vagner, A., Bogaerts, J.: A Messaging API for Inter-Widgets Communication. In: 18th WWW, pp. 1115–1116. ACM (2009)
15. Wilson, S., Daniel, F., Jugel, U., Soi, S.: Orchestrated User Interface Mashups Using W3C Widgets. Current Trends in Web Eng., pp. 49–61. Springer (2012)
16. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding mashup development. Internet Computing, IEEE, 12(5), 44–52 (2008)

Towards the automation of claiming in SLA-driven services

Manuel León, Pablo Fernandez, Manuel Resinas, and Antonio Ruiz-Cortés

University of Seville, Spain *
{mjleon,pablofm,resinas,aruiz}@us.es

Abstract. Current software industry is evolving into a service–centric scenario and consequently, the importance to create reliable service consumptions amongst organizations is a key point. In such a context, the concept of Service Level Agreement (SLA) represents the foundation to express the responsibilities (i.e. rights and obligations) of service consumer and provider during the consumption. However, in spite there has been a major effort in both academia and industry to develop languages and frameworks to support SLAs, there still remain important challenges to address such as how to automate the detection of a violation of the SLAs and how to react accordingly in order to claim for a compensation. Specifically, in this paper we focus on the definition of the automated claiming of SLAs problem characterized as the set of processes of gathering, checking and explaining the evidences associated with the service consumption within the context of an SLA. In order to identify the key requirements to automate the claiming of SLAs, we analyse the real case of the Simple Storage Service (S3) provided by Amazon, that is regulated by an SLA. Based on our analysis we propose a set of extensions to current prominent SLA language specification (WS–Agreement) and conceptualize a list of research challenges to automate the management of the claiming process.

1 Introduction

Service oriented systems is becoming the prominent paradigm in the software industry and, in such a scenario, the Service Level Agreement (SLA) represents a first-class citizen to describe for the rights and obligation of both service consumer and provider during the service consumption. Specifically, these concerns become a major issue in the current shift to the cloud, where a plethora of virtualized services (in the form of IaaS, PaaS and SaaS) are offered by different providers.

As a concrete motivational example we can analyse the real case of the Simple Storage Service (S3), provided by Amazon and regulated by an specific SLA

* This work was partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants P12-TIC-1867 (COPAS), TIN2012-32273 (TAPAS), TIC-5906 (THEOS))

[1] that is expressed in natural language. In this context, from the consumer perspective, detection of SLA violations is a key point to have a reliable usage and, in case of SLA violations, to exploit the appropriate compensations from the provider; consequently, we focus this paper in the problem of automating the SLA-driven claiming defined as the set of models and processes involving the gathering of potential evidences, SLA compliance checking, and violation explaining.

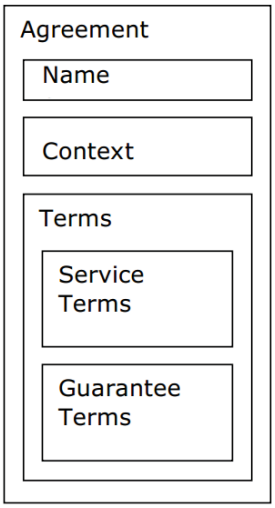

Fig. 1. WS-Agreement Structure

Specifically, in order to automate the claiming, we identify a twofold problem: firstly, a need for specifying the SLA with a formal model; secondly, a need for frameworks and tooling that support the automation of the involved processes.

In this problem context, some important steps have been developed: WS-Agreement specification[2] represents a prominent language and protocol for expressing SLAs in a formal way; this approach provides a framework that is extended with configurations which provide specific syntax for the terms and guarantees elements. Specifically, in [1] authors propose the iAgree language as a domain-independent set of extensions that complement WS-Agreement to develop a fully functional SLA language that can be applied in real scenarios such as the Amazon S3 SLA. However, to the best of our knowledge, currently there is a significant gap trying to exploit the SLA in order to support the claiming process.

Main contributions in this paper are twofold: (i) we propose an extension for WS-Agreement that provide the necessary placeholders for the automation of SLA-driven claiming; (ii) we identify a set of research challenges that can be used as requirements for a further design and development of tooling that support the automated process. In addition, as a minor contribution, we model Amazon S3 SLA with iAgree as a running example.

This paper is organized as follows: Section 2 presents WS-Agreement briefly. Section 3 details the case study that motivates this paper and the extension proposal of iAgree. Section 4 presents the identified research challenges. Finally, Section 5 shows some final remarks and conclusions.

[1] available at http://aws.amazon.com/es/s3/sla/

2 WS-Agreement in a nutshell

The WS-Agreement specification provides a document structure (see figure 2) to represent Service Level Agreements. An SLA defined in WS-Agreement must contains an agreement identifier, an agreement context and agreement terms.

The agreement context contains general information about the agreement, such as the responder of the agreement, the lifetime and the metrics data types and their domains.

The agreement terms are classified into two groups: service terms and guarantee terms. Service terms define the features of the service and are grouped in three sub-types depending on their goal: *Service references*, define the endpoint to the interface of the service; *Service descriptions*, define the features of the service agreed between consumer and provider. *Service properties*, specify the monitorable variables that will be used to define the guarantee terms. These properties must be specified in terms of the metrics available at the context. Guarantee terms (GTs) describe the SLOs that the responder, usually the provider, must fulfil. SLOs are assertions expressed using the service properties. A guarantee term can have a Qualifying Condition (QC) that determines when the GT is applicable or not. Penalty or reward clauses can appeal associated to a guarantee term. These clauses establish compensations and define the circumstances in which the consumer or the provider could apply for them.

It is important to highlight that WS-Agreement provide a generic framework that must be extended in order to address a particular scenario. Specifically, authors propose iAgree [1] a domain-agnostic extension to WS-Agreement that provides the specific sublanguages to express SDTs and GTs in a wide range of scenarios. Complementary, iAgree offers a human-oriented notation for WS-Agreements (originally described in XML) that is fully-compatible with the standard as long with a tooling[2] that can be used to write, transform (to/from XML) and analyse documents. For the sake of the readability, in the rest of the paper, we use the iAgree notation to express the SLA examples.

3 Case study

In order to ground our proposal in a real case, we analyse the Simple Storage Service (S3) provided by Amazon that is a highly popular service to persist objects in the cloud. This service provides an SLA[3] to describe the expected QoS (Quality of Service) that is expressed in natural language.

Specifically, the main QoS property is the *Monthly Uptime Percentage* (MUP, see figure 2), calculated by subtracting from 100% the average of *error rates*—number of requests resolved as *Internal Error* or *Service Unavailable* divided by the total number of requests during a five minute period, expressed as a percentage—measurements during the billing cycle. Amazon S3 guarantees an MUP greater or equal than 99.9%

[2] Available at http://labs.isa.us.es/apps/iAgreeStudio/
[3] http://aws.amazon.com/es/s3/sla/

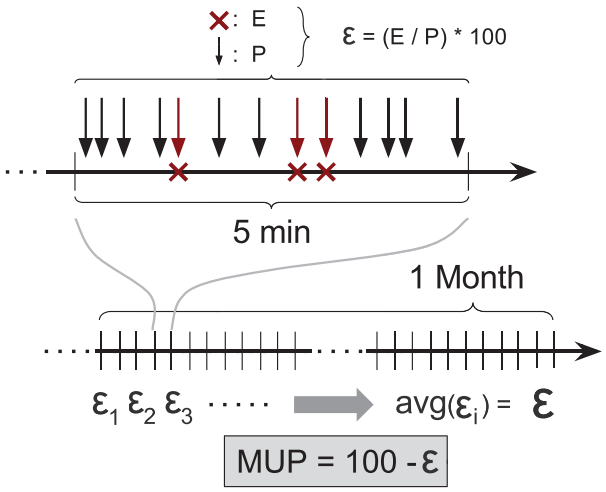

Fig. 2. Monthly Uptime Percentaje (MUP) calculation

In case Amazon fails to meet this MUP, service consumers are able to apply for a compensation, as long as they can provide evidence of violation. This compensation is expressed as a factor called Service Credit Percentage (SCP) that is applied, as a discount, in the following billing cycle.

Using the actual version of WS-Agreement, a first approach to formalize this SLA is shown in figure 3. In this iAgree document we can outline all the different elements described in the natural language version; however, there is a significant gap over the actual definition of the MUP measurement that is mandatory to gather the appropriate evidences to proof an SLA violation.

In this context, it is important to notice that this gap represents a relevant constraint over the potential infrastructure of evidence gathering; specifically, the lack of the appropriate information in the SLA impose a rigid structure for evidence gatherers since they have to include a hard-wired implementation of the measurements (i.e. MUP). In addition, this coupling has proved to be highly inconvenient since Amazon (amongst other companies) is known to frequently change the SLA (including in some cases an actual MUP measurement procedures redefinition): provider could change the SLA and redefine the QoS and the gatherer would not notice it.

Another important limitation of this SLA definition is the lack of time scopes: The SLA should contain the information needed to express clearly required properties for real scenarios such as the lifetime (when it starts and expires) of the SLA; time scopes for metrics; and time periods for guarantee terms. This drawback is specially important in real scenarios like Amazon S3, where this information could determine if an SLA can be considered violated or fulfilled, or when start to consider an SLA fulfilled again after a violation.

IAgree

```
AgreementOffer AmazonS3SLA version 1.0 for AmazonS3 version 1.0
     Provider as Responder
     Metrics:
        percent: float [0..100]
        regions: enum {us-east-1, us-west-1, us-west-2, eu-west-1,
        ap-southeast-1, ap-southeast-2, ap-northeast-1, sa-east-1}

AgreementTerms
     Service AmazonS3 availableAt. "http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl"

     GlobalDescription
        authorizationKey: string = "yourAWSAccessKeyId";
        region: regions = "us-west-1";
        SCP: percent = 0; //Service Credit Percentage: discount calculated by current month
                          //billing applicable to the next billing cycle.

     MonitorableProperties
        global:
           MUP: percent

     GuaranteeTerms
        G1: Provider guarantees MUP >= 99.9;
            with monthly penalty
                of SCP=10 if MUP >= 99 && MUP <99.9;
                of SCP=25 if MUP < 99;
            end

EndAgreementOffer
```

Fig. 3. Amazon S3 SLA expressed in IAgree language

3.1 Our proposal

Our contribution is centred on improving the SLA-driven claiming automation by decoupling the metrics semantic from the monitors. Specifically, we propose a twofold extension to WS-Agreement:

- For each service (SDT), add an endpoint to the service that will monitor the SLA. Including the monitor in the SLA, allow a dynamic binding that can be evolved through the SLA lifecycle.
- For each of the measurements required in the domain, add an endpoint to a specialized metering agent.

Figure 4 exemplify this extension in our use case of Amazon S3. In this SLA, the MUP is defined as a metric. Note that MUP property actually represents the concept "availability" of a service as Amazon defines it. Alternatively, other providers define availability with different expressions, but they usually represent a similar concept. Moreover, MUP has a different meaning within the context of provider Amazon depending on the service (cf. EC2). Such a variability is conveniently handled by the usage of the metric concept.

The set of proposed extensions boost a decoupled deployment architecture that has the SLA as the centric element to base all the domain-specific parameterization. Specifically, (as shown in figure 5), we envision a potential infrastructure divided in the following elements:

IAgree extended

```
AgreementOffer AmazonS3SLA version 1.0 for AmazonS3 version 1.0
     Provider as Responder
     Metrics:
        MUP: "http://labs.isa.us.es/apps/MUPMeter"
        percent: float [0..100]
AgreementTerms
     Service AmazonS3 availableAt "http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl"
     Monitor HTTPMOnitor availableAt "http://labs.isa.us.es/apps/HTTPMonitor?SLAid=id"

     GlobalDescription
        SCP: percent = 0;

     MonitorableProperties
        global:
           Availability: MUP
     GuaranteeTerms
        G1: Provider guarantees Availability >= 99.9;
            with monthly penalty
                of SCP=10 if Availability >= 99 && Availability <99.9;
                of SCP=25 if Availability < 99;
            end
EndAgreementOffer
```

Fig. 4. Amazon S3 SLA expressed in IAgree extended language

- Monitor. This element is responsible for notifying the metering agents with the occurring events (event log).
- Metering agent. It calculates an specific metric based on the events provided by the monitor.
- Claim manager. Based on the measurements and the SLA, it analyse and detect violations; in such a case it will gather the appropriate evidences to support a claim to the provider.

4 Research challenges

Based on the analysis of our case of study and the proposed extension of WS-Agreement, in this section we identify challenges to be address in the development of a SLA-driven infrastructure for automated claiming. These challenges are based on a potential engineering of SLA claiming processes by monitoring the SLA, analysing the violations and gathering evidences required to claim for penalties (or rewards) expressed in the SLA:

- *Monitorable QoS properties measurement.* Coding ad-hoc meters for the monitorable properties can be a tedious and error-prone task, specially for complex expressions like MUP for Amazon S3.
  Challenge 1: Provide a set of composable meters that offers the needed functionality to measure complex expressions and generate automatically their configuration from an SLA.
- *SLA checking level.* Currently, most SLA monitors and analysers are able to check if the SLA is violated; but they cannot explain which guarantee

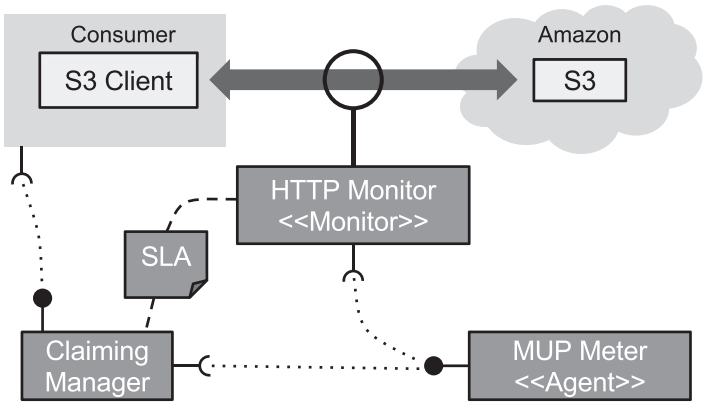

Fig. 5. Potential automated SLA claiming infrastructure

terms were transgressed. In our opinion, the checking level should be the guarantee term because the clauses applicable in an eventual violation are linked directly to them.

Challenge 2: Monitor each guarantee term as an independent entity, and explain violations for guarantee terms instead for SLOs or SLAs.

– *Time scopes.* SLAs, as any contract, have a valid period time. Also, the guarantee terms or the metrics could have a time scope, e.g. Amazon S3 MUP is calculated monthly.

Challenge 3: Extend current SLA tooling to support time scopes for every level: SLA, guarantee terms and metrics.

– *Penalties and rewards.* WS-Agreement establish penalty and reward clauses in the SLA. It would be interesting to provide mechanisms to calculate the clauses that clients could demand.

Challenge 4: Provide support for automatic claiming process for penalties and rewards, based on the monitoring and analysis results and gathering the required proofs of violations.

5 Conclusions

Automating the claim of SLAs in real services is a challenging problem. In spite there are formal languages to express SLAs (such as WS-Agreement), they fail to include the necessary information to automate gathering evidences and checking violations in order to have a solid claim that could lead to the actual compensation. In this paper, we propose a WS-Agreement extension (grounded in iAgree) that includes the necessary placeholders to evolve the SLA document as the key placeholder that parametrize an automated claiming infrastructure. Moreover, as a case study, we exemplify our proposal by analysing and developing a formal SLA of the Simple Storage Service (S3) provided by Amazon. Finally, based on

this analysis, we identified the following research challenges: (i) to automate the generation of meters for the QoS properties; (ii) monitor guarantee terms rather than SLAs; (iii) supports time scopes for SLAs, guarantee terms and metrics; and (iv) calculate penalties and rewards automatically based on the monitoring results. These challenges could act as the design principles to create further flexible infrastructures that provide an automated support for SLA-based claiming.

References

1. Müller, C., Gutiérrez, A.M., Resinas, M., Fernández, P., Ruiz-Cortés, A.: iagree studio: A platform to edit and validate WS-Agreement documents. In: 11th International Conference on Service Oriented Computing (ICSOC). (2013)
2. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical report, Global Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG (2011)
3. Müller, C., Oriol, M., Frach, X., Marco, J., Resinas, M., Ruiz-Cortés, A., Rodríguez, M.: Comprehensive explanation of SLA violations at runtime. IEEE Transactions on Services Computing (2013)

Metaherramienta para la generación de aplicaciones científicas basadas en *workflows*

Rubén Salado-Cid, José Raúl Romero y Sebastián Ventura

Dpto. de Informática y Análisis Numérico
Universidad de Córdoba, Campus de Rabanales, 14071 Córdoba
{rsalado,jrromero,sventura}@uco.es

Resumen. El uso de la programación visual en el ámbito científico ha contribuido al desarrollo de aplicaciones que facilitan la realización de experimentos. Actualmente, existen aplicaciones para trabajar sobre un único dominio, limitadas a procedimientos propios de ese dominio, y aplicaciones multidominio, cuya complejidad para la configuración de sus elementos supone un gran esfuerzo para el usuario. Por tanto, es necesario ofrecer flexibilidad para trabajar sobre diversos dominios y permitir una adaptación intuitiva a dominios conocidos por el usuario. En este trabajo se presenta una metaherramienta para la generación automática de aplicaciones adaptadas a un dominio, o conjunto de dominios, para la composición y ejecución de *workflows* científicos en términos de procesos locales y servicios remotos. Estas nuevas aplicaciones disponen de una infraestructura que proporciona interoperabilidad con aplicaciones externas, presentan interfaces de usuario personalizadas y abstraen al usuario final de la complejidad de configuración al ofrecer elementos de trabajo ya adaptados a su campo.

Palabras clave: workflow científico, metaherramienta, programación visual, servicios remotos

1. Introducción

Actualmente, el uso de aplicaciones basadas en *workflows* está muy extendido, con un especial crecimiento en el ámbito científico en la última década. Este tipo de aplicaciones, gracias a la utilización del paradigma de la programación visual [14], permite a un usuario la definición y gestión de algoritmos sin la necesidad de ser experto en programación. Para su representación visual se dispone de una interfaz gráfica de usuario que, haciendo uso de diversos elementos visuales, proporciona acceso a una serie de procesos o recursos (de forma local o remota), aislando al usuario final de los aspectos de implementación internos, como el lenguaje de programación utilizado o la plataforma subyacente.

Un objetivo común de estos sistemas es el de facilitar a científicos un acceso transparente a numerosos procesos, herramientas y recursos útiles en un determinado dominio, o conjunto de dominios, para la realización de experimentos y el posterior análisis de los resultados obtenidos. La complejidad de los problemas

abordados en estos experimentos hace necesario que expertos de distintos ámbitos trabajen conjuntamente para alcanzar una solución final. Sin embargo, la diversidad de tecnologías utilizadas en el desarrollo de este tipo de aplicaciones, como el modelo de computación subyacente, el lenguaje de *workflow* utilizado o el entorno de ejecución, hace que lograr este objetivo no sea algo trivial [8].

Dada la gran diversidad de ámbitos científicos (bioquímica, genética, biomedicina, etc.), han surgido una serie de herramientas multidominio [6,12,15] que ofrecen distintos elementos configurables para acceder a recursos y procesos propios de cada dominio. A pesar de la versatilidad que presentan este tipo de aplicaciones, su complejidad y el tiempo de diseño aumenta con respecto a aplicaciones más específicas debido a la constante necesidad de configuración de parámetros, adecuación de procedimientos, etc. Así pues, la complejidad y la falta de adaptación del entorno de trabajo al dominio específico, hace que sea necesario buscar mecanismos que ofrezcan flexibilidad para trabajar sobre múltiples dominios, a la vez que permitan al usuario una adaptación y parametrización más intuitiva para los dominios que le resultan familiares.

En este artículo se presenta una metaherramienta para la creación de aplicaciones científicas basadas en *workflows*, orientada a dos perfiles diferenciados de usuario. Los usuarios con perfil avanzado son los encargados de personalizar y generar las nuevas aplicaciones específicas para cada dominio. Para ello, se ofrecen mecanismos que les permiten crear y adaptar una serie de elementos, como procesos, fuentes de datos, servicios remotos o visualizadores de datos, que formarán parte de las aplicaciones generadas. Por otra parte, los usuarios con perfil de experto en el dominio son los usuarios finales de dichas aplicaciones, que no necesariamente requieren conocer su estructura interna o un lenguaje de programación. Las aplicaciones generadas presentan una interfaz personalizada según los criterios del desarrollador, ofrecen todos los elementos funcionales básicos para la composición de *workflows*, permiten la ejecución de los *workflows* definidos tanto con la propia aplicación como con otras aplicaciones generadas con la metaherramienta, y proporcionan un acceso directo a los procesos, servicios y recursos adaptados al dominio, abstrayendo al usuario experto de la complejidad de su configuración. Además, se ha desarrollado un lenguaje específico de dominio (DSL, *Domain-Specific Language*) para la descripción de los *workflows*, cuya sintaxis concreta es interpretada por el motor de ejecución de la herramienta. Para lograr la integración con herramientas externas, su sintaxis abstracta, descrita en términos del metamodelo, permite obtener una versión compatible con otros lenguajes de *workflows* mediante transformaciones de modelos.

En el resto del artículo se presenta en profundidad la metaherramienta propuesta. En la Sección 2 se muestra el trabajo relacionado, incluyendo algunas de las herramientas más utilizadas en este ámbito. En la Sección 3 se presenta el DSL desarrollado para la ejecución de los *workflows*. En la Sección 4 se detalla la arquitectura general y los módulos de la metaherramienta. La Sección 5 expone un caso de estudio realizado para la generación de una aplicación sobre un dominio específico. Finalmente, en la Sección 6 se presentan las conclusiones y se explican algunas líneas de trabajo futuro.

2. Trabajo relacionado

Los sistemas basados en *workflows* han sido utilizados tradicionalmente para la automatización de los procesos de negocio [2]. Los lenguajes utilizados en este ámbito [11] usan flujos de control para especificar el orden de realización de las tareas definidas dentro del propio *workflow*. Esta tecnología ha sido adoptada por la comunidad científica para la automatización de los experimentos [1], si bien presentan diferencias notables con los primeros [18]. Debido al uso intensivo de los datos que se lleva a cabo durante un experimento, los lenguajes utilizados para la definición de *workflows* científicos utilizan flujos de datos para definir el orden de ejecución de sus elementos [19].

Dentro del ámbito científico, se encuentran disponibles numerosas aplicaciones basadas en *workflows* que permiten trabajar sobre dominios específicos, como minería de datos [3,7], bioinformática [16], neurociencia [13], etc. También se han desarrollado herramientas independientes del dominio, como Taverna [15], Kepler [12] o Triana [6]. Taverna permite la definición y ejecución de *workflows* científicos proporcionando acceso a recursos locales y remotos, así como a herramientas de análisis. Se integra con otras herramientas como myExperiment [17], para el intercambio de *workflows*, y BioCatalogue [4], para el descubrimiento de servicios web para el ámbito científico. Kepler es una herramienta diseñada para ayudar a científicos, analistas y programadores a desarrollar, ejecutar e intercambiar modelos y análisis en numerosos ámbitos científicos, cuya principal característica es el modelado orientado a actores [5]. Triana es un entorno para la resolución de problemas, que proporciona herramientas para el análisis de datos y cuenta con un gran número de componentes para trabajar en dominios como el procesamiento de audio, texto, etc. La principal limitación de estas herramientas radica en la complejidad para ser adaptadas a un dominio específico, y en la imposibilidad de generar aplicaciones particularizadas con elementos de trabajo ya configurados. Por ello, resultaría de interés contar con una metaherramienta en el campo de los *workflows* científicos que facilitara esta adaptación.

De hecho, ya se han realizado esfuerzos para facilitar el metadesarrollo de aplicaciones en otros ámbitos. En el contexto de MDE (*Model Driven Engineering*), por ejemplo, existen herramientas como GMF [10], para la generación de editores gráficos a partir de la definición del modelo de dominio que determina sus características, o Xtext [9], para el desarrollo de IDEs (*Integrated Development Environment*) de lenguajes específicos del dominio, cuya infraestructura incluye analizadores, enlazadores o intérpretes del lenguaje. Hasta donde sabemos, no existen metaherramientas que permitan la generación automática de aplicaciones de gestión de *workflows* científicos adaptadas a dominios específicos.

3. Lenguaje específico para la ejecución de *workflows*

La descripción a alto nivel de los procesos, servicios y recursos definidos en un *workflow* debe ser convertida en una versión ejecutable por la plataforma. Para especificar de forma precisa toda la información relevante para la ejecución

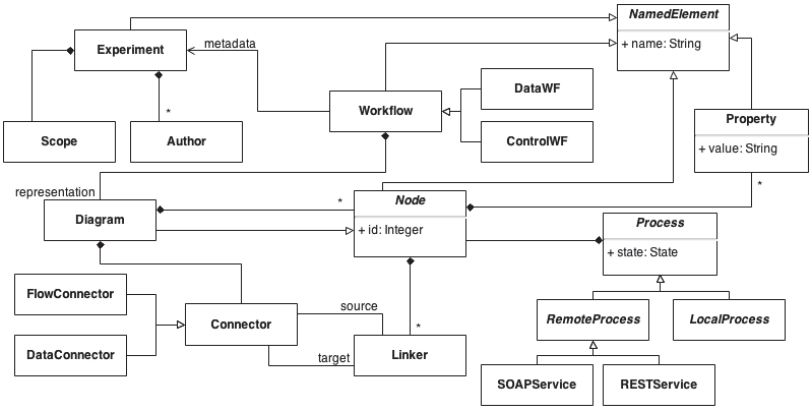

Fig. 1. Vista parcial simplificada de la sintaxis abstracta del lenguaje.

de *workflows*, se ha modelado un DSL propio, cuya sintaxis abstracta ha sido definida en términos de meta-modelos. Por motivos de espacio, la Figura 1 muestra parcialmente el meta-modelo que define el lenguaje, el cual está dividido en diferentes paquetes, según la especialización y funcionalidad de sus elementos: *workflow*, entrada y salida de datos, procesos y estructuras de control.

Workflow. Define la metainformación del experimento (*Experiment*) y del *workflow* a ejecutar. Soporta dos tipos, dependiendo de si es dirigido por flujos de control (*ControlWF*) o por flujos de datos (*DataWF*). Su representación es definida mediante un diagrama (*Diagram*), que cuenta con una serie de nodos (*Node*) y conectores (*Connector*).

Entrada y salida de datos. Incluye los elementos que definen las fuentes de datos heterogéneas, locales y remotas, de entrada y salida. Cada elemento está compuesto por un único enlazador (*Linker*), que representa un punto de interacción en una conexión de flujo de datos (mediante *DataConnector*) para la incorporación, visualización o almacenamiento de datos.

Proceso. Incluye la declaración de todo procedimiento, local o remoto, que manipula unos datos de entrada para generar datos de salida. Un procedimiento local (*LocalProcess*) es aquel que se ejecuta en la propia máquina donde se encuentra la aplicación. Puede ser de tres tipos, según su alcance y funcionalidad: proceso complejo (transforma un conjunto de entradas en un conjunto de salidas mediante un procedimiento complejo), diagrama (*workflow* que define un procedimiento anidado) y filtro (únicamente dedicado a la manipulación sencilla de los flujos de datos como, por ejemplo, realizar conversiones de tipos). Un procedimiento remoto (*RemoteProcess*) define un servicio externo, al que actualmente se puede acceder mediante REST o SOAP/WSDL. Cada elemento está compuesto por tantos enlazadores como datos de entrada y de salida defina el procedimiento, y permite tanto conexiones de flujo de control (*FlowConnector*) como de datos.

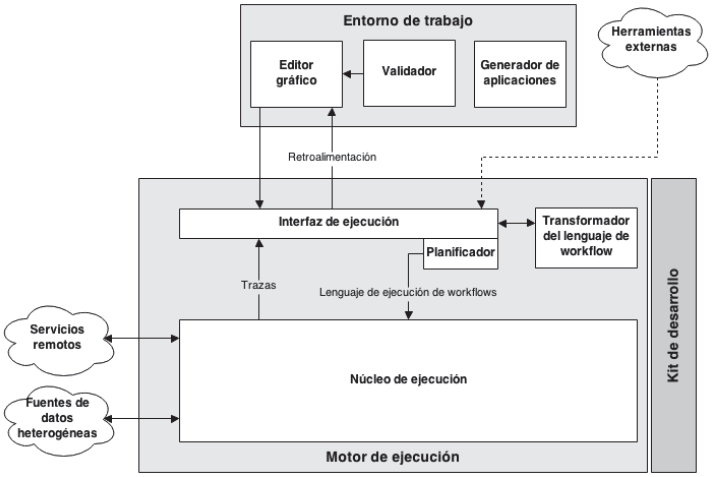

Fig. 2. Arquitectura general de la metaherramienta.

Estructuras de control. Incluye los elementos que determinan el flujo de ejecución. Se consideran dos tipos de estructuras de control: esquemas condicionales y bucles. Los esquemas condicionales definen las salidas activas tras la evaluación de una condición. Pueden ser de tipo *si-entonces* (*if-else*) y de tipo *en-caso-de* (*switch*). Un bucle define el valor de una variable que, junto al uso de esquemas condicionales, se utiliza para la declaración de iteraciones.

4. Arquitectura general de la herramienta

La Figura 2 muestra el esquema de la arquitectura general de la metaherramienta. Es una arquitectura modular que separa sus componentes en dos elementos principales: el entorno de trabajo (que permite al usuario la composición de procesos y definición de *workflows*, además de la configuración, personalización y generación de las nuevas aplicaciones) y el motor de ejecución (que procesa los *workflows* compatibles con la plataforma). A este módulo se accede además mediante un kit para desarrolladores que permite extender su funcionalidad e interoperabilidad con herramientas y lenguajes externos.

4.1. Entorno de trabajo

El entorno de trabajo, que ha sido desarrollado como *plugin standalone* de la plataforma Eclipse, posee una interfaz gráfica de usuario que consta a su vez de distintos componentes. Para la creación de *workflows* se ofrece un editor gráfico, que dispone de los elementos necesarios para su composición, definición y control de ejecución. Para el desarrollo y creación automática de las nuevas aplicaciones se ofrecen un conjunto de *wizards* y mecanismos para la selección de recursos, procesos y servicios que formarán parte de las mismas.

Editor gráfico. Está compuesto por una paleta de herramientas y un área de composición de diagramas, orientada al experto en el dominio, para la creación visual de *workflows* complejos y jerárquicos. A través de esta paleta es posible seleccionar cada uno de los nodos y conectores, tal y como se definen en el DSL, que conformarán el *workflow*. El editor también permite la configuración de cada elemento y ofrece información precisa sobre su ejecución y depuración.

Validador. Es el módulo encargado de validar en vivo el *workflow* durante la fase de creación del mismo. Comprueba que se cumplen las restricciones impuestas por la notación abstracta del lenguaje y notifica al usuario sobre posibles errores cometidos en la composición y configuración de los elementos que lo conforman.

Generador de aplicaciones. Permite la generación automática de aplicaciones específicas, particularizadas con elementos de trabajo previamente adaptados al dominio. Para ello, se debe introducir la información relativa a estos elementos, junto con la información necesaria para personalizar su interfaz gráfica. Estas nuevas aplicaciones, exportadas como *plugin standalone* de Eclipse, proporcionan un entorno para la composición y ejecución de *workflows*, con acceso a procedimientos ya adaptados al dominio, tanto visual como funcionalmente.

4.2. Motor de ejecución

Se ha desarrollado un motor de ejecución propio que se encarga de invocar servicios y recursos remotos, ejecutar procedimientos locales, interpretar datos de entrada y salida, así como de calcular el flujo de ejecución y de emitir trazas. Además, permite que la ejecución de los *workflows* pueda ser realizada tanto de forma regular como en modo depuración.

Interfaz de ejecución. El motor de ejecución gestiona la comunicación con módulos externos, ofreciendo flexibilidad y permitiendo así la escalabilidad de la herramienta. Esta interfaz es la encargada de recibir las peticiones de ejecución de *workflows* y de proporcionar la retroalimentación requerida.

Transformador de lenguajes. A partir de la sintaxis abstracta del DSL, permite realizar transformaciones entre lenguajes de *workflows*. Así, permite la interoperabilidad con aplicaciones externas basadas en *workflows* al permitir ejecutar otras notaciones concretas desde el módulo de ejecución, independientemente de la interfaz de usuario utilizada.

Planificador. Realiza un procesamiento previo sobre la representación a alto nivel de los procedimientos y recursos definidos en el *workflow* para obtener una versión ejecutable del mismo. Este procesamiento trata de optimizar la ejecución teniendo en cuenta distintos aspectos, como el tipo de *workflow* definido (dirigido por flujos de control o por flujo de datos), la cantidad de memoria disponible, las unidades o nodos de ejecución de la plataforma, la tipología de dicha plataforma de ejecución, etc.

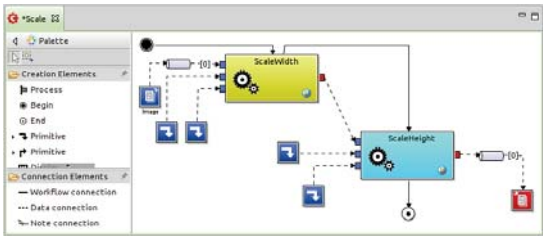

Fig. 3. Flujo de control sencillo invocando a servicios remotos.

Núcleo de ejecución. A partir de una descripción ejecutable de un *workflow* obtenido por el planificador, este componente ejecuta cada uno de sus elementos según las pautas establecidas. Durante este proceso, se generará la información sobre la ejecución en forma de trazas, junto con los resultados del experimento.

Kit de desarrollo. Para facilitar la extensibilidad tanto del motor de ejecución como de la herramienta, se han desarrollado una serie de interfaces de programación (actualmente en Java), que permiten a un desarrollador incrementar la variedad de tipos de datos con los que puede trabajar la herramienta, ampliar múltiples funcionalidades propias del sistema, así como extender el catálogo disponible de filtros, procesos, visualizadores de datos, etc.

5. Generación de una herramienta para el procesamiento de imágenes: caso de estudio

Para mostrar el funcionamiento de la metaherramienta a la hora de generar nuevas aplicaciones específicas a un dominio, se trabajará sobre un caso de estudio focalizado en la manipulación de imágenes. Esta nueva aplicación generada contará con una serie de procedimientos, desarrollados mediante programación visual, abstrayendo al usuario final de toda la complejidad de diseño y configuración, y que le permitirán realizar tratamientos específicos sobre imágenes de entrada, invocando servicios remotos para realizar su procesamiento.

En primer lugar, se deberán crear (o bien buscar ya existentes) y configurar los procesos que formarán parte de la nueva aplicación, mediante el editor gráfico. En la Figura 3 se muestra la paleta con los elementos del lenguaje disponibles, así como un *workflow* ya definido para modificar el tamaño de una imagen de entrada, que contiene dos procesos remotos que invocan a un servicio web, cuyas entradas y salidas son descubiertas automáticamente por la herramienta y dispuestas en el diagrama (analizando el WSDL en el caso de los servicios SOAP, y a partir de la URL en el caso de los servicios REST), y dos filtros para adaptar el tipo de dato al requerido por el servicio y por el elemento de salida, respectivamente. El resto de elementos se corresponden con las imágenes de entrada y salida, con cuatro datos de entrada de usuario referidos a las coordenadas de escalado de la imagen, y con los puntos de inicio y fin del *workflow*. Durante esta

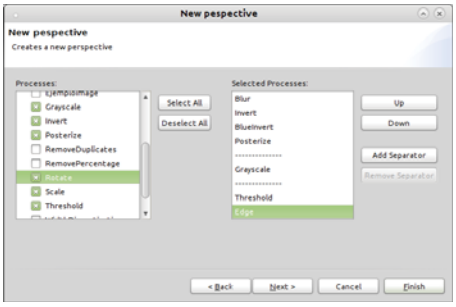

Fig. 4. Configuración y selección de procesos (ventana de diálogo).

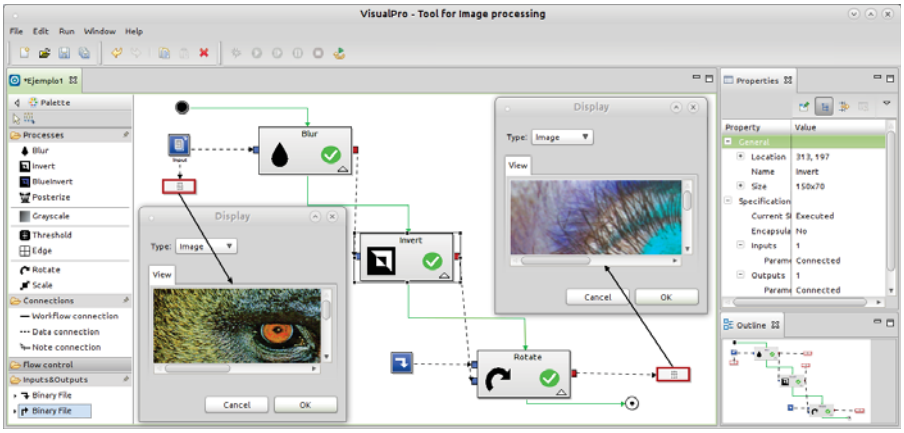

Fig. 5. Nueva herramienta generada para la manipulación de imágenes.

fase, el usuario dispone del kit de desarrollo para extender la funcionalidad de la metaherramienta incorporando nuevos procesos locales, si fuera necesario.

Una vez creados todos los *workflows* correspondientes a los procedimientos del dominio, se configurará la nueva aplicación a través del componente generador de herramientas. Para ello, el usuario deberá introducir información relativa a la interfaz gráfica, como el nombre de la nueva aplicación, una breve descripción de la misma y otra metainformación. Igualmente, se deberá proveer información sobre los procedimientos creados y configurados anteriormente, ya adaptados al dominio (Figura 4). Tras indicar estos elementos, y una vez personalizada y parametrizada la interfaz, se procede a la generación automática de la aplicación. Como se observa en la Figura 5, el entorno generado y los procesos disponibles, así como la descripción de los elementos y el tipo de *workflow*, se ha personalizado al dominio concreto, ocultando los detalles de implementación al usuario final. De esta manera, se ofrece una aplicación más intuitiva en su uso y adaptable al tipo de experimento objetivo.

6. Conclusiones y trabajo futuro

En este trabajo se ha descrito la primera metaherramienta para la generación de aplicaciones basadas en *workflows* en el ámbito científico. Las herramientas más genéricas existentes no abstraen al usuario final de la complejidad existente a la hora de extender su funcionalidad, o de configurar y adaptarla a un dominio concreto. Por tanto, con esta metaherramienta y su orientación a distintos perfiles de usuario se consigue mitigar este problema de cara al usuario final, proporcionando los mecanismos para crear, configurar y generar nuevas aplicaciones adaptadas al dominio. Para la definición y ejecución de los *workflows* se ha modelado un DSL cuya sintaxis abstracta permite la transformación desde y hacia otros lenguajes de *workflows*, permitiendo la interoperabilidad con otras herramientas. La arquitectura de la plataforma da soporte a estas funcionalidades presentando una estructura modular y extensible, ofreciendo además un kit de desarrollo para que programadores externos puedan ampliar y adaptar múltiples funcionalidades del sistema. Finalmente, para ilustrar el funcionamiento de esta herramienta, se ha mostrado un caso de estudio sencillo, en el que se ha generado una aplicación basada en *workflows* para un dominio específico. La aplicación resultante presenta una interfaz de usuario personalizada por el desarrollador, junto con los recursos y procesos basados en servicios seleccionados previamente que permiten al usuario experto trabajar sobre su propio dominio.

La línea a seguir en el trabajo futuro de la herramienta consiste en dotarla de un mayor número de mecanismos para la personalización de las aplicaciones generadas. Se desarrollará un catálogo de transformaciones de modelos para ofrecer compatibilidad con otros lenguajes de *workflows*. También se trabajará en la integración del motor de ejecución con otras plataformas tecnológicas (p. ej. computación distribuida, *grid computing*, etc.) para aumentar su rendimiento.

Agradecimientos

Trabajo apoyado por el Ministerio de Ciencia y Tecnología, proyecto TIN2011-22408, y fondos FEDER.

Referencias

1. Akram, A., Meredith, D., Allan, R.: Evaluation of BPEL to scientific workflows. In: Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on. vol. 1, pp. 269–274 (May 2006)
2. Barker, A., Hemert, J.: Scientific workflow: A survey and research directions. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol. 4967, pp. 746–753. Springer Berlin Heidelberg (2008)
3. Berthold, M., Cebron, N., Dill, F., Gabriel, T., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications, pp. 319–326. Studies in Classification, Data Analysis, and Knowledge Organization, Springer (2008)

4. Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., Wolstencroft, K., Aleksejevs, S., Stevens, R., Pettifer, S., Lopez, R., Goble, C.A.: BioCatalogue: a universal catalogue of web services for the life sciences. Nucleic Acids Research 38(Web-Server-Issue), 689–694 (2010)
5. Bowers, S., Ludäscher, B.: Actor-oriented design of scientific workflows. In: Proc. of the International Conference on Conceptual Modeling (ER). pp. 369–384 (2005)
6. Churches, D., Gombás, G., Harrison, A., Maassen, J., Robinson, C., Shields, M.S., Taylor, I.J., Wang, I.: Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience 18(10), 1021–1037 (2006)
7. Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From experimental machine learning to interactive data mining. In: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) Knowledge Discovery in Databases: PKDD 2004, Lecture Notes in Computer Science, vol. 3202, pp. 537–539. Springer (2004)
8. Elmroth, E., Hernández, F., Tordsson, J.: Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. Future Generation Comp. Syst. 26(2), 245–256 (2010)
9. Eysholdt, M., Behrens, H.: Xtext: Implement your language faster than the quick and dirty way. In: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion. pp. 307–309. SPLASH '10, ACM (2010)
10. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional, 1 edn. (2009)
11. Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.): Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007, Innsbruck, Austria, June 7, 2007, CEUR Workshop Proceedings, vol. 251. CEUR-WS.org (2007)
12. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)
13. MacKenzie-Graham, A., Payan, A., Dinov, I., Horn, J., Toga, A.: Neuroimaging data provenance using the LONI Pipeline workflow environment. In: Freire, J., Koop, D., Moreau, L. (eds.) Provenance and Annotation of Data and Processes, Lecture Notes in Computer Science, vol. 5272, pp. 208–220. Springer (2008)
14. Myers, B.: Taxonomies of visual programming and program visualization. Journal of Visual Languages and Computing 1(1), 97–123 (1990), cited By (since 1996)78
15. Oinn, T., Greenwood, M., Addis, M., Ferris, J., Glover, K., Goble, C., Hull, D., Marvin, D., Li, P., Lord, P.: Taverna: Lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006)
16. Rampp, M., Soddemann, T., Lederer, H.: The MIGenAS integrated bioinformatics toolkit for web-based sequence analysis. Nucleic Acids Research 34(Web-Server-Issue), 15–19 (2006)
17. Roure, D.D., Goble, C., Bhagat, J., Cruickshank, D., Goderis, A., Michaelides, D., Newman, D.: myExperiment: Defining the social virtual research environment. In: 4th IEEE International Conference on e-Science. pp. 182–189. IEEE Press (2008)
18. Yildiz, U., Guabtni, A., Ngu, A.: Business versus scientific workflows: A comparative study. In: Services - I, 2009 World Conference on. pp. 340–343 (July 2009)
19. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. Journal of Grid Computing 3(3-4), 171–200 (2005)

Methodology to Extend RAL*

Cristina Cabanillas[1], Manuel Resinas[2],
Antonio Ruiz-Cortés[2], and Jan Mendling[1]

[1]Vienna University of Economics and Business, Austria
{cristina.cabanillas,jan.mendling}@wu.ac.at
[2]University of Seville, Spain {resinas,aruiz}@us.es

Abstract. Resource Assignment Language (RAL) is a language for the selection of organisational resources that can be used, for example, for the assignment of human resources to business process activities. Its formal semantics have allowed the automation of analysis operations in several phases of the business process lifecycle. RAL was designed considering a specific organisational metamodel and pursuing specific purposes. However, it can be extended to deal with similar problems in different domains and under different circumstances. In this paper, a methodology to extend RAL is introduced, and an extension to support another organisational metamodel is described as a proof-of-concept.
Keywords: business process management, description logics, RAL, resource assignment, W3C Organisation Ontology

1 Introduction

Resource Assignment Language (RAL) is a language for the assignment of performers to Business Process (BP) activities. Its formal semantics based on Description Logics (DLs) enable the automation of analysis operations in several phases of the BP lifecycle, so that questions such as who can take part in a specific BP activity, or who is never involved in a certain BP, can be automated.

RAL was developed to address specific gaps existing in Business Process Management (BPM). Specifically, it was designed following a rationale grounded on an already defined organisational metamodel and on the Workflow Resource Patterns (WRPs) [1], which capture the way in which resources can be managed in BPs. Thus, one of the limitations of the language is that it can only be used in organisations that implement certain organisational structures supported by the metamodel. However, RAL was designed as a modular language to make it extensible, due to several reasons, a.o.: (i) the selection of human resources might be required in other domains or for further purposes, e.g., to select people for the composition of teams; (ii) different organisations might have different organisational structures, which, besides, might evolve over time; (iii) the selection of non-human resources might also be useful in BPM and other domains.

* This work was partially supported by the European Union's Seventh Framework Programme (FP7/2007-2013), the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants 318275 (GET Service), P12-TIC-1867 (COPAS), TIN2012-32273 (TAPAS), TIC-5906 (THEOS)).

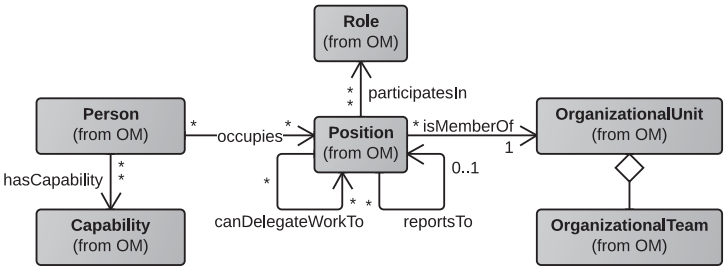

Fig. 1: Excerpt of the organisational metamodel described by Russell et al. [1]

In this paper, a methodology to extend RAL is introduced, describing guidelines and tips on how to perform each step. In addition, as a proof-of-concept of the application of the methodology, the language is extended to support the World Wide Web Consortium (W3C) Organisation Ontology (OO) in order to show how it can be made compatible with other organisational metamodels.

The rest of this paper is structured as follows. In Section 2, RAL is outlined. In Section 3, the methodology to extend it is described. In Section 4, a concrete example of the implementation of the methodology is detailed. Finally, in Section 5 the conclusions drawn from this work are presented.

2 Background. RAL

Resource Assignment Language (RAL) is a Domain Specific Language (DSL) explicitly developed to define conditions to select which resources can be potential participants of a BP activity, i.e., to assign resources to activities. It was first introduced in [2], extended in [3] and revised in [4]. Its specification and its implementation within Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL) can be found at `http://www.isa.us.es/cristal/`.

The language allows formulating expressions that make reference to elements of an organisational model and to elements of a BP model. In particular, in the current version, the organisational metamodel has to (at least partially) implement the organisational metamodel described by Russell et al. in [1], which consists of persons, capabilities, positions, roles and organisational units, as depicted in Fig. 1. The BP metamodel used in RAL is highly inspired in Business Process Model and Notation (BPMN) [5], abstracting the most important characteristics of the current approaches of such types of metamodels and adapting some parts for resource analysis purposes. With these elements, RAL specification and semantics were defined as summarised next.

2.1 RAL Specification

RAL is a modular language composed of *expressions* that may contain different types of *constraints*. It comprises RAL Core, which allows defining basic resource

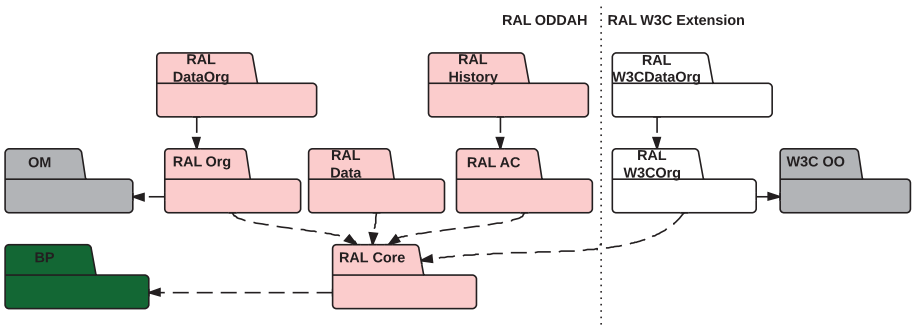

Fig. 2: RAL ODDAH and extension to support the W3C Organisation Ontology

assignments; and several extensions that add new types of expressions and/or constraints. In particular, RAL ODDAH is made up of the modules depicted on the left hand side of Fig. 2[1] and outlined next[2]. Their Extended Backus-Naur Form (EBNF) syntax can be found in [4].

RAL Core contains four types of expressions (*PersonExpr*, *IsAssignmentExpr*, *NegativeExpr*, *CompoundExpr*) and one type of constraint (*PersonConstraint*) to define conditions based on people's characteristics. Among others, it allows specifying conditions such as `IS David`, and `(NOT (IS Anna)) AND (NOT (IS Alex))`.

RAL Org extends RAL Core with four types of expressions (*GroupResourceExpr*, *CommonalityExpr*, *CapabilityExpr*, *HierarchyExpr*) to select people according to their organisational information on the ground of the metamodel depicted in Fig. 1, and four types of constraints (*PositionConstraint*, *RoleConstraint*, *UnitConstraint*, *CapabilityConstraint*) to tune the selection conditions. It allows specifying conditions such as `HAS ROLE PhDStudent`, `HAS CAPABILITY Degree`, and `CAN DELEGATE WORK TO POSITION HrmsPhDStudent`, among others.

RAL Data *and* ***RAL DataOrg*** allow selecting individuals or group resources[3] indicated in a data field, by extending *PersonConstraint* (*RAL Data*), and *PositionConstraint*, *RoleConstraint* and *UnitConstraint* (*RAL DataOrg*). They allow specifying conditions like `IS PERSON IN DATA FIELD Form.Purchaser`, and `IS UNIT IN DATA FIELD Application.DepartmentName`.

RAL AC stands for RAL Access-Control and it extends *PersonConstraint* to enable the specification of constraints related to the resources allocated to

[1] OM and BP represent the organisational and the BP metamodels used in RAL, respectively.
[2] Due to space restrictions, the extension of RAL ODDAH described in Section 4 is also depicted in Fig. 2.
[3] *Group resource* is used to jointly refer to positions, roles and organisational units.

other activities, specifically Binding of Duties (BoD) or Segregation of Duties (SoD) with the actual performer of another activity of *the same* BP instance, e.g., `IS ANY PERSON responsible for ACTIVITY SubmitCRV`, and `NOT (IS ANY PERSON accountable for ACTIVITY SignAuthorisation)`.

RAL History extends RAL AC's *PersonConstraint* to enable referencing previous execution instances of a BP, giving rise to conditions like `IS ANY PERSON accountable for ACTIVITY SubmitCRV IN ANOTHER INSTANCE`, and `IS ANY PERSON responsible for ACTIVITY BuyTickets FROM 01/01/2014 TO 01/07/2014`.

The RAL modules can be composed with each other to define more complex conditions. For instance, `(HAS UNIT Hrms) AND (SHARES SOME POSITION WITH ANY PERSON responsible for ACTIVITY SubmitCRV)` combines RAL Org and RAL AC. In order to make RAL expressive, the design goal was to provide as much support as possible for the following selection criteria: (i) the relationships represented in the organisational model of the company; (ii) the Creation Patterns, a subset of the WRPs [6] that capture behaviour related to resource selection; and (iii) the different degrees of responsibility a person can have for an activity, i.e., the *task duties* associated to the activities. Task duties have been considered in recent releases of languages such as WS-HumanTask [7] and BPEL4People [8], as well as in the so-called RACI matrices [9]. In RAL, they are introduced in RAL AC.

As a result, RAL provides capabilities to define all the relationships of an organisational model that implements totally or partially the metamodel used in the language, full support for the Creation Patterns, and the opportunity to assign resources with different degrees of responsibility to a single activity.

2.2 RAL Semantics

RAL semantics were defined considering the formalisation principles defined in [10]. The formalisation is based on a semantic mapping to DLs [11] with the primary objective of establishing a sound basis for sophisticated automated support, supported by two reasons. First, RAL expressions can be seen as a way to specify a subset of the people of an organisation by defining conditions they must satisfy. This way of defining RAL expressions fits nicely into the way DLs express their concepts and, hence, they provide a very natural way to describe the problem, which allows following the *Semantics Priority Principle* and makes it easier to avoid unnecessary representational choices as suggested by the *Conceptualisation Principle*. As a consequence, we can define RAL Core semantics, and then extend them for RAL Org, RAL Data, RAL DataOrg, RAL AC and RAL History without modifying the essence. The second reason is that there is a plethora of off-the-shelf DL reasoners that can be used to automatically analyse RAL expressions and, thus, to automatically infer information from them.

In order to formalise RAL using DLs, every model related to RAL has to be mapped into DL elements either in the TBox or in the ABox of a Knowledge Base (KB). Although there is a significant number of concepts in the problem domain, we tried to keep the number of concepts in the formalisation as small as

possible, as suggested by the *Orthogonality Principle*. The detailed description of RAL semantics can be found in [12].

3 Methodology to Extend RAL

Two different approaches can be followed to extend RAL. The first one is applicable when a new organisational metamodel needs to be used, and consists of mapping the concepts and relationships of such metamodel to the ones depicted in Fig. 1, and then use RAL as defined in Section 2. Query View Transformation (QVT) techniques could be used to this end. However, this option may not be feasible if the new organisational metamodel highly differs from the one used in RAL. If this is the case, or the purpose of the extension is to apply RAL to a different domain, the second option should be used, which consists of extending RAL specification and semantics as is explained next.

3.1 Define Expressions and Constraints

First of all, RAL specification has to be extended. For instance, in case of extending RAL to support a new organisational metamodel, new concepts and new organisational relationships may be considered to define resource selection conditions. We next describe the principles taken into account for the design of RAL expressions and constraints, which should be used as guideline for extensions:

1) *Keep the EBNF specification compact.* The goal is to learn to use the language by following similar patterns for the definitions of the selection conditions. For instance, for all the organisational entities, the pattern `HAS + orgEntityName + id` has been used, so that the user can define conditions such as `HAS ROLE Researcher`, `HAS POSITION HrmsClerk` and `HAS UNIT Hrms`.
2) *Keep syntax close to natural language.* The aim is to increase understandability and readability.
3) *Keep the minimal complete subset of coherent selection conditions.* For that purpose, it is also necessary to consider the rationale used to define RAL expressions and constraints (cf. Section 2). In our design of RAL, this implied, e.g., limitting the types of expressions that can be negated. As a consequence, we found expression type `NOT (IsAssignmentExpr)` very generic, and expression type `NOT (CompoundExpr)` unnecessarily hard to understand and automate; hence, they were left aside the language specification.
4) *Keep a distinction between expressions and constraints.* An expression is a type of selection conditions, whereas a constraint specifies restrictions that can be defined for an expression type, i.e., variations regarding the conditions bounded to a type of selection expression. In general, all the conditions related to an expression type begin the same way or follow some pattern, but specific parts can be configured with constraint types for different selection criteria. Therefore, constraints are defined to be reused in different expression types. For instance, the pattern shown in the previous point is

used for the expression type *GroupResourceExpr*. Another example is pattern `IS PersonConstraint` defined for *PersonExpr*, which leads to expressions such as `IS David` and `IS PERSON IN DATA FIELD Form.Purchaser`; the former specifies a particular person and belongs to RAL Core, and the latter indicates that the person is written on a field of a data object and belongs to RAL Data. *PersonConstraint* is, in turn, used in other expression types, e.g., *CommonalityExpr*.

5) *Keep module decomposition reasonable.* New RAL expressions and constraints may give rise to new RAL modules, whose scope must be well defined.

It may not be possible to meet 100% of the five aforementioned recommendations simultaneously, so a balance among them should be pursued, instead. For instance, the syntax of the language may be affected by a compact EBNF specification, but reasonable readability levels must be ensured. In the extension presented in Section 4, we follow the order proposed above. However, it may change depending on the purpose of the extension. For instance, if it is intended to be used by developers mainly, the second guideline might not be so important.

3.2 Define DL-Based Semantics

RAL DL-based KB must be extended by adding elements to the TBox and the ABox, and doing all the configurations required following a similar procedure as the one used to define RAL semantics, which is summarised next.

1) *Define a mapping from the expressions of the extension to DLs.* Since all expressions define resource selection conditions, the new RAL expressions must be mapped to DL concepts that are subconcepts of Person so that all people included in a DL concept meet the conditions of an expression.
2) *Define a mapping from the constraints of the extension to DLs.* The new RAL constraints must be mapped to DL concepts according to the type defined by the constraint. For instance, a role constraint would be mapped as a subconcept of DL concept Role.

In addition, if the expressions and constraints introduced in the extension refer to concepts of a metamodel that has not been included by other RAL extension yet, it is necessary to map them according to the following guidelines.

3) *Map the new metamodel(s) used by the extension in the TBox.* This involves mapping the classes and relationships of the new metamodel(s) into concepts and properties in the TBox, respectively, together with the required configuration, e.g., cardinality restrictions.
4) *Map the instances of the metamodel(s) in the ABox.* This involves mapping the elements of specific models into individual assertions in the ABox as instances of the concepts introduced in the previous step.
5) *Configure the KB with additional DL axioms.* Specifically, DL axioms must be added to deal with the *open world assumption*, which involves defining every concept and property as different from each other with the *disjoint with* axiom, and stating that there are no relationships between individuals other than those explicitly defined.

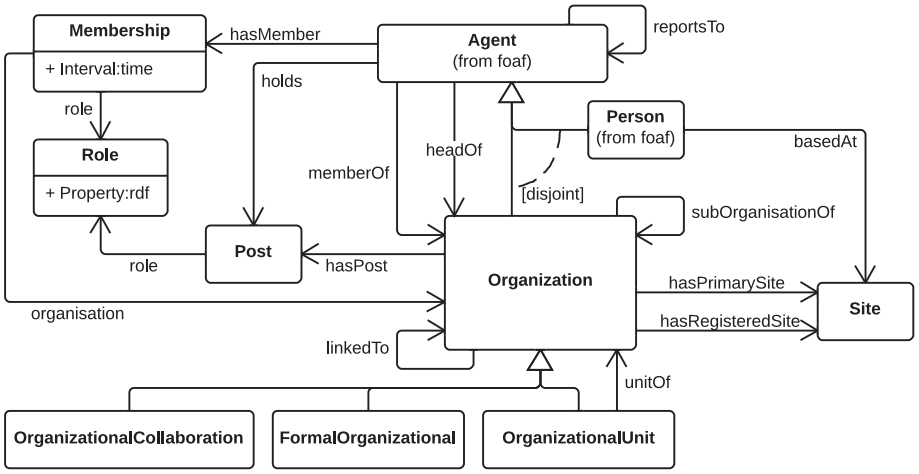

Fig. 3: W3C Organisation Ontology

4 Using RAL with the W3C Organisation Ontology

As a proof-of-concept, in the following we describe the extension of RAL to make it compatible with the OO defined by the W3C as recommendation to represent organisational structures[4], whose metamodel is depicted in Fig. 3.

In short, an *Agent* (an *Organisation* or a *Person* as defined in another W3C ontology) is member of one or more organisations that she can head. The metamodel is intended to be used in many different types of organisations, so particular sub-classes of organisation are distinguished. Organisations can be structured hierarchically and exist in locations (class *Site*). The primary site indicates the default means by which an organisation can be contacted, and the registered site indicates a legally registered site for the organisation. A person is also located at a specific site. Agents can play *Roles* in organisations during a specific period of time, which is represented with class *Membership*. Roles are described by a set of properties that define the associated rights and duties. Furthermore, a set of positions (class *Post*) can be defined as basis to define the reporting line. Unlike roles, positions exist regardless of whether they are currently held by some agent or not. The W3C OO also includes the concept of organisation evolution, but it is disregarded in this section due to space limitation.

4.1 Adding New Expressions and Constraints to RAL

The extension of RAL specified in Language 1 has been defined following the principles described in Section 3.1. It is the result of adding two new modules (cf. right hand side of Fig. 2). RAL W3COrg enables assignments based on the

[4] `http://www.w3.org/TR/vocab-org/`

Language 1 Specification of modules RAL W3COrg and RAL W3COrgData

```
1  RALExpression := ... | W3CExpr
2
3  PersonConstraint := personName
4                    | LOCATED [IN | AT] address
5
6  W3CExpr := W3CGroupResourceExpr
7           | W3CReportingExpr
8
9  W3CGroupResourceExpr := IS HEAD OF OrgConstraint
10   | HAS (OrgConstraint | PostConstraint)
11   | (HAS | HAD) RoleConstraint
12
13 W3CReportingExpr := REPORTS TO (PersonConstraint | PERSON WHO W3CExpr)
14   | IS REPORTED BY (PersonConstraint | PERSON WHO W3CExpr)
15
16 OrgConstraint := orgName
17   | [OrgType] [WITH SiteType LOCATION address]
18         [WHICH IS A (SUBORGANISATION | SUPERORGANISATION) OF orgName]
19   | OrgType IN DATA FIELD dataObject.fieldID
20
21 PostConstraint := POST (positionName | WITH ROLE roleName+
22   | IN DATA FIELD dataObject.fieldID) IN OrgConstraint
23
24 RoleConstraint := ROLE (roleName
25   | IN DATA FIELD dataObject.fieldID) IN OrgConstraint [TimeInterval]
26
27 OrgType := ORGANISATION | ORGANISATIONAL COLLABORATION
28   | FORMAL ORGANISATION | ORGANISATIONAL UNIT
29
30 SiteType := PRIMARY | REGISTERED | λ
```

W3C OO (cf. Fig. 3) by extending RAL Core with a new type of expression (*W3CExpr*, line 1) that involves two subtypes of expressions, and a new *PersonConstraint*[5] that allows selecting people depending on their location (line 4). RAL W3CDataOrg extends RAL W3COrg to support Deferred Allocation (cf. WRPs [6]). The expressions and constraints of both modules are described next:

- **W3CGroupResourceExpr (line 6):** It involves all the conditions for the selection of people according to characteristics of the group resources they are related to, i.e., (i) the organisations they head (line 9), (ii) the organisations they are member of (line 10), (iii) the positions they hold (line 10), and (iv) the roles they play (line 11). Notice that, in order to keep the EBNF specification compact and, thus, ease learnability, the pattern `HAVE + restOfExpr` has been used instead of each specific relationship of the metamodel. That pattern could not be applied to point (i), but the name of the relationship defined in the organisational metamodel has been used, instead.
- **W3CReportingExpr (line 7):** It allows defining restrictions according to the reporting line defined in the organisation (lines 13-14), either by specifying a person given by a *PersonConstraint*; or by defining a new restriction with the help of the W3C OO extension itself.

[5] *PersonConstraint* was introduced in RAL Core and extended in RAL Data and RAL AC [4].

– **OrgConstraint:** It allows defining restrictions with regard to the organisations of which the potential performers are member. Specifically, RAL W3COrg allows specifying (i) a concrete organisation (line 16); and (ii) an organisation of a determined or undetermined type with specific characteristics regarding its location (line 17), and its hierarchical relationship with other organisations (line 18). RAL W3CDataOrg allows indicating that the organisation of which the potential performers are member are specified in a data field of a BP (line 19). Note that relationship *linkedTo* of the W3C OO has been excluded following principle 3 defined in Section 3.1, since it is a very generic relationship for which applicability has not been found without performing a refinement, e.g., to provide details about the connection between the organisations. However, that is out of the scope of this paper.
– **PostConstraint:** It allows defining restrictions with regard to the positions held by the potencial performers (line 21), specifically a position in particular, or a position related to a specific (set of) role(s) in an organisation, in the case of RAL W3COrg; and a position indicated in a data field within an organisation, in the case of RAL W3CDataOrg.
– **RoleConstraint:** It allows defining restrictions with regard to the roles played by the potential performers in an organisation for a specific period of time, specifically a concrete role, in the case of RAL W3COrg (line 24); and a role indicated in a data field, in the case of RAL W3CDataOrg (line 25).

4.2 Extending RAL Semantics

Since the expressions and constraints of the extension refer to elements of a new metamodel, the W3C OO, the extension of the RAL semantics involves mapping the metamodel together with the expressions and constraints. According to Section 3.2, three steps are necessary to include the metamodel in the DL-based KB. However, in this case the first and second steps are not necessary since the W3C OO is already defined as a DL-based KB itself. Therefore, only the step to deal with the open world assumption is necessary. For instance, if there is an Agent a that just holds one Post p, it is necessary to add to the DL-based KB that $holds(a, p)$ and that $\{a\} \sqsubseteq \forall holds.(\{p\})$, i.e., that agent a only holds position p.

Concerning the mapping of the expressions and constraints, it is necessary to define a mapping μ from expressions and constraints to DL concept descriptions. Next, we detail some of them.

$$\mu(\texttt{IS HEAD OF OrgConstraint}) \equiv \exists headOf.(\mu(OrgConstraint))$$
$$\mu(\texttt{HAS RoleConstraint}) \equiv \exists hasMember.(\exists role.(\mu(RoleConstraint)))$$
$$\mu(\texttt{POST WITH ROLE r IN OrgConst}) \equiv \exists role.(\{r\}) \sqcap \exists hasPost^{-}.(\mu(OrgConst))$$

The first two mappings map expressions included in *W3CGroupResourceExpr*. Note that the DL concepts in which the expressions are mapped are subconcepts of Person and both of them use the mapping again to map the constraints referenced in the expression. The third mapping maps a *PostConstraint* into a DL concept that is subconcept of Post.

5 Conclusions

In this paper, a methodology to extend RAL has been introduced. RAL is a language for the selection of human resources to be assigned to BP activities, which may require an adaptation or extension to be used for other purposes. As an example of the use of the methodology, RAL has been extended to support a new organisational metamodel, specifically the W3C OO. Extensions to apply RAL to other domains could also be useful, e.g., in the context of document management in order to define permissions for data access, thus replacing the BP-related part of RAL for document management aspects.

The aim of this work is to make it easier for any user to design and use ad-hoc versions of RAL following the same rationale and principles used to define the specification and semantics of the language. RAL is currently being extended applying such methodology for the selection of teams for collaborative work.

References

1. N. Russell, A. ter Hofstede, D. Edmond, and W. M. P. van der Aalst, "Workflow Resource Patterns," tech. rep., BETA, WP 127, Eindhoven University of Technology, 2004.
2. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes," in *Business Process Management Workshops (BPD'11)*, pp. 50–61, 2011.
3. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Designing Business Processes with History-Aware Resource Assignments," in *BPM 2012 Workshops (BPD'12)*, vol. 132, pp. 101–112, 2012.
4. C. Cabanillas, M. Resinas, A. del Río-Ortega, and A. Ruiz-Cortés, "Automated Design-Time and Run-Time Analysis of the Business Process Resource Perspective," *Information Systems*, p. Under review, 2013.
5. OMG, "BPMN 2.0," recommendation, OMG, 2011.
6. 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.
7. "Web Services-Human Task (WS-HumanTask) v1.1," tech. rep., OASIS, 2010.
8. "WS-BPEL Extension for People (BPEL4People)," tech. rep., OASIS, 2009.
9. M. Smith, "Role And Responsibility Charting (RACI)," in *Project Management Forum (PMForum)*, p. 5, 2005.
10. A. H. M. T. Hofstede and H. Proper, "How to Formalize It? Formalization Principles for Information System Development Methods," *Information and Software Technology*, vol. 40, pp. 519–540, 1998.
11. B. Motik and R. Rosati, "Reconciling description logics and rules," *J. ACM*, vol. 57, pp. 30:1–30:62, June 2008.
12. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Defining and Analysing Resource Assignments in Business Processes with RAL," in *ICSOC*, vol. 7084, pp. 477–486, 2011.

Una solución basada en HTCondor para aprovechar la disponibilidad de recursos efímeros

Sergio Hernández, Javier Fabra, Joaquín Ezpeleta, Pedro Álvarez

Instituto de Investigación en Ingeniería de Aragón (I3A)
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
{shernandez,jfabra,ezpeleta,alvaper}@unizar.es

Resumen Clusters, grids y, más actualmente, clouds, representan las infraestructuras de computación dedicada más utilizadas por científicos e investigadores. Sin embargo, existen otras alternativas cuya principal función no es la computación y que pueden ser útiles para la resolución de problemas computacionalmente costosos. En el ámbito académico existen una gran cantidad de recursos que, durante gran parte del día, permanecen encendidos y desaprovechados, de forma que se podrían utilizar para tareas computacionales durante el tiempo que permanecen infrautilizados. Con este objetivo, en este artículo se propone la utilización del *middleware* HTCondor para aprovechar estos *recursos efímeros* existentes en nuestro departamento, así como su integración en un *framework* de computación distribuida desarrollado anteriormente y que en la práctica estamos utilizando y extendiendo para resolver problemas complejos. Esta integración se ha realizado mediante la adición de un componente en el *framework* que, basándose en el horario de reserva de los recursos integrados, es capaz de recomendar los más adecuados para la ejecución de cada trabajo. Finalmente, se ha utilizado el entorno de Amazon EC2, simulando el entorno real de ejecución, para configurar la nueva infraestructura y probar el nuevo componente.

Palabras clave: Recursos efímeros, computación en la nube, planificación de recursos, heterogeneidad, HTCondor.

1 Introducción

La investigación en ciertas áreas científicas tiene unos requisitos computacionales muy elevados [1]. Como respuesta a estas necesidades, se han utilizado infraestructuras de computación de altas prestaciones que proporcionan al usuario un elevado número de recursos, como ocurre en el caso de los *clusters* y los *grids* [2]. Recientemente, se ha propuesto el uso de la virtualización para aumentar las capacidades de este tipo de infraestructuras, dando lugar a la aparición del *cloud computing* [3, 4]. Como punto en común, estas infraestructuras proporcionan un elevado conjunto de recursos dedicados de forma exclusiva a tareas de computación.

Sin embargo, tanto en los entornos académicos como laborales existen una gran cantidad de recursos que no se utilizan durante gran parte del tiempo que permanecen encendidos (ordenadores de trabajo del personal, laboratorios de docencia, etc.). De esta forma, durante el tiempo en el que los recursos están infrautilizados, pueden ser usados para ejecutar tareas computacionales sin que afecte a los usuarios de dichos recursos. Estos recursos, denominados *recursos efímeros*, pueden ser utilizados como una infraestructura de computación alternativa que podría sustituir o complementar a los clusters, grids y clouds [5].

Para aprovechar este tipo de recursos, existen diferentes *middleware* de gestión. Uno de los más conocidos es BOINC [6], una plataforma que permite utilizar ordenadores distribuidos por todo el mundo a través de Internet para ejecutar aplicaciones pertenecientes a proyectos científicos. Sin embargo, BOINC está orientado a la ejecución de aplicaciones muy específicas en entornos no confiables. Una alternativa que extiende BOINC añadiendo la posibilidad de organizar los recursos jerárquicamente es el SZTAKI *Desktop Grid* [7]. Este sistema incluye una versión capaz de trabajar en entornos donde los recursos son confiables pero sigue permitiendo tan sólo la ejecución de unas pocas aplicaciones. Otro de los *middlewares* más utilizados es HTCondor [8]. Se trata de un sistema de computación de altas prestaciones que permite aprovechar los ciclos libres de ordenadores desaprovechados de una organización. Su principal ventaja con respecto a los sistemas anteriores es la posibilidad de ejecutar cualquier tipo de trabajo. Finalmente, existen plataformas como OpenStack [9], OpenNebula [10] o Eucalyptus [11] que permiten crear un *cloud* privado proporcionando máquinas virtuales para ejecutar trabajos. Sin embargo, este tipo de soluciones están orientadas a entornos donde los recursos están dedicados.

El objetivo de este artículo consiste en explotar la disponibilidad de los recursos efímeros en nuestro entorno de trabajo para su utilización futura en la resolución de problemas computacionalmente costosas. Para ello, se va a utilizar el *middleware* HTCondor y se va a integrar la nueva infraestructura en un *framework* de computación distribuida, que integra diferentes infraestructuras cluster, grid y cloud, desarrollado previamente por los autores [12]. Para la integración, se propone el desarrollo de un nuevo componente que sea capaz de utilizar información sobre la reserva de los recursos integrados para guiar el proceso de asignación de trabajos a recursos, mejorando la utilización y el aprovechamiento de los mismos. Finalmente, cabe destacar que un elemento clave para esta integración es la utilización de un entorno de computación en la nube, como es el caso de Amazon EC2 [13], para simular el entorno real y facilitar la implementación y configuración de la infraestructura y del nuevo componente.

El resto del artículo se organiza de la siguiente forma. En la sección 2 se describe el *framework* de computación y el entorno de recursos efímeros disponible. La sección 3 muestra el diseño del nuevo componente encargado de gestionar la infraestructura de recursos efímeros. En la sección 4 se detalla cómo se ha llevado a cabo la implementación de la infraestructura, así como su validación utilizando la infraestructura cloud de Amazon. Finalmente, la sección 5 concluye el artículo y reflexiona sobre las futuras líneas de trabajo.

2 Antecedentes

En esta sección se presenta el *framework* desarrollado por los autores y se muestra cómo encaja la nueva infraestructura de recursos efímeros en el mismo. Adicionalmente se presentan las características más relevantes de la nueva infraestructura y de su implementación utilizando HTCondor.

2.1 Descripción del *framework* de computación distribuida

En [12] presentamos un *framework* de computación orientado a servicios para el despliegue y ejecución de *workflows* científicos en entornos de computación heterogéneos. Este *framework* integra y gestiona de forma transparente un amplio conjunto de recursos de computación y los ofrece al usuario como un único y potente entorno de ejecución. La figura 1 muestra el diseño arquitectural del *framework*, formado por tres capas. La *capa de interfaz de usuario* permite programar las aplicaciones a ejecutar mediante diferentes lenguajes y herramientas. La *capa de ejecución* es la responsable de gestionar el despliegue de las aplicaciones en las infraestructuras de computación integradas. Para ello, incluye dos tipos de componentes que se comunican a través de un *bus de mensajes*: los *componentes de gestión*, que gestionan el ciclo de vida de las aplicaciones; y los *mediadores*, que interactúan con cada infraestructura concreta a través de su *middleware*. Finalmente, la *capa de infraestructuras de computación* engloba los recursos de ejecución utilizados. Actualmente se han integrado un clúster (HERMES) y dos grids (AraGrid y PireGrid), pertenecientes a nuestra Universidad, y la infraestructura cloud de Amazon [13].

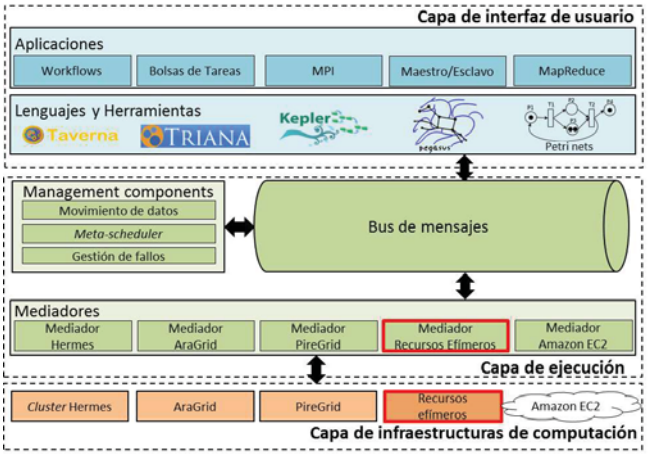

Figura 1: Diseño arquitectural del framework de comptutación.

Como puede observarse en la figura, la integración de esta nueva infraestructura afecta tanto a la capa de ejecución como a la de infraestructuras de computación. Por un lado, debe desarrollarse un nuevo mediador capaz de interactuar con la infraestructura de recursos efímeros. Por otro lado, debe implementarse

la nueva infraestructura de recursos efímeros mediante la instalación y configuración del *middleware* HTCondor en nuestro entorno de trabajo.

En lo referente a este último punto, la figura 2 muestra la topología del escenario de recursos efímeros. Como puede observarse, está compuesta por un servidor, que integrará los servicios de gestión necesarios en HTCondor, y varios laboratorios de docencia de nuestro departamento, en los que se ejecutarán las tareas computacionales. En total, se utilizan 154 recursos de ejecución con un total de 350 procesadores y 494 GB de RAM de los cuales calculamos que cada hora hay aproximadamente un 35% disponibles.

Para la implementación de HTCondor en el entorno disponible se ha utilizado el servidor de gestión para alojar las funciones de gestión (*Central Manager* y Condor Connection Broker, CCB) y envío de trabajos (*Submit*), mientras que los recursos de ejecución albergan la función de ejecución (*Execute*) y también permiten el envío de trabajos de forma remota (*Submit remote*). La figura 3 muestra dicho diseño indicando las funciones configuradas en cada recurso.

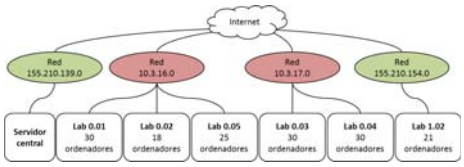

Figura 2: Topología de red de la infraestructura de recursos efímeros.

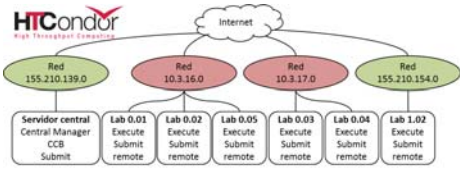

Figura 3: Funciones de HTCondor en la infraestructura de recursos efímeros.

3 Diseño de un mediador de gestión de recursos efímeros

Para integrar la infraestructura de recursos efímeros en el *framework* anterior, se ha programado un nuevo mediador capaz de gestionar este tipo de recursos e interactuar con HTCondor. Este mediador es capaz de obtener los trabajos a ejecutar procedentes del *framework*, someter los trabajos para su ejecución, monitorizar el estado de los recursos y los trabajos en ejecución, recuperar los resultados de los trabajos finalizados y proporcionárselos a los usuarios a través del *framework* y gestionar el horario de ocupación de los recursos.

La idea general y principal contribución de este nuevo mediador es la de utilizar información acerca de la reserva de los recursos para guiar el proceso de planificación (asignación de trabajos a recursos computacionales). Esto es posible ya que, en este caso, los recursos integrados se utilizan para la realización de prácticas por los alumnos por lo que se dispone de información sobre los horarios de reserva de dichos recursos. El nuevo mediador utiliza esta información para seleccionar los recursos más adecuados para la ejecución de cada trabajo de acuerdo a su disponibilidad. En cualquier caso, no se descarta la ejecución de trabajos en recursos que estén reservados si el resto de recursos están ocupados o no disponibles. Esta decisión se debe a que el hecho de que un laboratorio esté reservado no implica que todos sus recursos vayan a ser ocupados y, por tanto, es posible que haya recursos libres que puedan ser utilizados para ejecutar tareas.

Adicionalmente, se plantea la utilización de información histórica que complemente a la información sobre la ocupación de los recursos. En cuanto a este tipo de información estamos interesados en guardar información tanto de los trabajos ejecutados (duración, recurso de ejecución, número de expulsiones, etc.) como de los propios recursos (tiempo medio libre y ocupado, instantes en los que el recurso pasa de estar libre a ocupado y viceversa, etc.). Con este registro histórico, se pretende poder mejorar en un futuro el proceso de planificación.

La figura 4 muestra el diseño arquitectural del mediador de recursos efímeros. En la misma, los componentes que afectan de forma exclusiva a la gestión de recursos efímeros aparecen resaltados. Antes de detallar los aspectos más relevantes referentes a estos componentes, vamos a describir el ciclo de vida base de los trabajos (correspondiente únicamente a los componente sin resaltar):

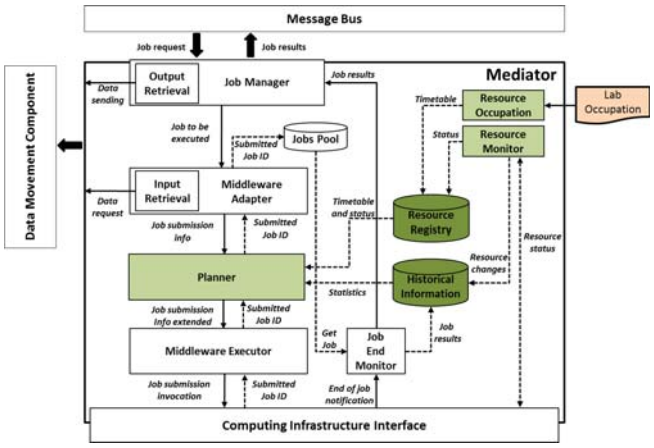

Figura 4: Diseño arquitectural del mediador de gestión de recursos efímeros.

En primer lugar, el *Job Manager* obtiene los trabajos a ejecutar del *framework* a través del *Message Bus* y los envía al *Middleware Adapter*. Este componente los traduce de un lenguaje de descripción estándar utilizado por el *framework* a un lenguaje que sea interpretable por el *middleware* de gestión de la infraestructura, en este caso HTCondor; y se encarga de mover a la infraestructura los datos de entrada necesarios para la ejecución del trabajo. Una vez realizada dicha traducción y que se han movido los datos de entrada necesarios, el trabajo es enviado al *Middleware Executor* (el uso del *Planner* es, por tanto, opcional) y el identificador asignado por la infraestructura es almacenado en el *Jobs Pool* junto con la descripción del trabajo. Cuando el trabajo finaliza su ejecución, el *Job End Monitor* lo detecta, consulta el *Jobs Pool* y se lo indica al *Job Manager* el cual recupera los resultados del trabajo y los envía al *framework* vía el *Message Bus* para que se pueda notificar al usuario.

Una vez mostrado el ciclo de vida del mediador, describimos de forma individual los componentes dedicados a la gestión específica de recursos efímeros:

– *Resource Monitor*: este componente monitoriza de forma periódica el estado de los recursos (libre, ocupado, eliminado, etc.).

- *Resource Occupation*: este componente es el encargado de gestionar la información sobre la ocupación de los recursos. Recibe un fichero de texto con información de la ocupación de los recursos, obtenido de la página web de nuestro departamento, lo traduce y lo almacena en el *Resource Registry*.
- *Resource Registry*: este almacén de información se encarga de gestionar la información referente a los recursos. Contiene el estado actual de los recursos (procedente del *Resource Monitor*) e información acerca de su ocupación (proporcionada por el *Resource Ocuppation*).
- *Historical Information*: este almacén de información se encarga de almacenar los logs con información histórica. Contiene información sobre los trabajos ejecutados (procedente del *Job End Monitor*) y eventos referentes a la adición/eliminación de los recursos integrados en el entorno de computación (procedente del *Resource Monitor*) . También permite obtener estadísticas que puedan utilizarse durante la planificación.
- *Planner*: este componente permite seleccionar el recurso o grupo de recursos en los que se recomienda ejecutar un trabajo. El componente recibe la información del trabajo a someter y le añade el recurso o grupo de recursos recomendados. Para ello, puede utilizar diferentes fuentes de información como son: el estado actual de los recursos y la ocupación prevista para los mismos (obtenidos del *Resource Registry*) y estadísticas de uso (proporcionadas por la componente *Historical Information*). Actualmente sólo utiliza información acerca de la ocupación de los recursos.

4 Validación de la solución utilizando recursos cloud

En esta sección se describe cómo se ha utilizado el entorno de computación en la nube de Amazon para implementar y configurar la infraestructura de recursos efímeros. Además, se presenta una prueba de concepto que simula una ejecución real con el objetivo de validar el mediador desarrollado y mostrar sus ventajas.

La razón por la que se ha utilizado un entorno virtual, como Amazon EC2 para la implementación, configuración y validación de la solución es doble: por una parte, evita una serie de problemas que nos hubiésemos encontrado si hubieses utilizado directamente el entorno real, como problemas de acceso a los recursos, falta de permisos de administración, disponibilidad limitada de los recursos, pérdida de control en la configuración del sistema, etc.; por otra parte, proporciona ventajas adicionales, ya que facilita la administración del sistema y de los recursos, la configuración de los mismos, la definición y validación de diferentes escenarios de error, el acceso a los recursos en todo momento, etc.

4.1 Despliegue y configuración de HTCondor

Para el despliegue y configuración del *middleware* HTCondor, que gestiona la infraestructura de recursos efímeros, se ha definido un entorno simulado lo más parecido al entorno real mostrado en la figura 3. Esto simplifica la labor de implantar posteriormente el sistema en el entorno real ya que, una vez

configurado y validado el entorno simulado, sólo es necesario replicar dicha configuración en el entorno real realizando cambios mínimos en los ficheros de configuración. De esta forma, la figura 5 muestra la topología del entorno simulado creada en Amazon EC2. Como puede observarse, al comparar este escenario con el escenario mostrado en la figura 3, ambos escenarios son análogos.

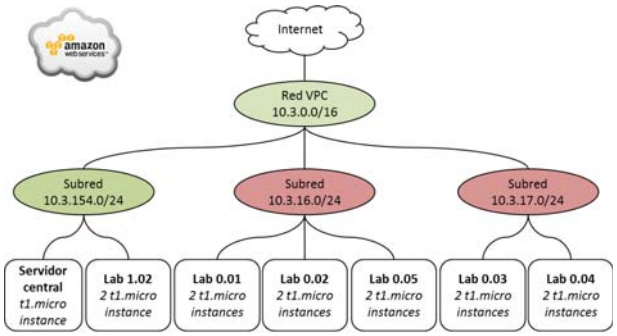

Figura 5: Topología del escenario de recursos efímeros simulado en Amazon EC2.

El aspecto más importante en la definición del escenario simulado es la topología de red, ya que se deben distinguir y separar las redes públicas de las privadas. Para simular la configuración de red se ha utilizado la función VPC (*Virtual Private Cloud*) de Amazon que permite crear redes virtuales privadas, con varias subredes y una puerta de enlace con salida a Internet. Utilizando esta función y configurando adecuadamente los grupos de seguridad (similar a un *firewall*) de las máquinas creadas, se ha replicado una topología de red como la existente en nuestro departamento universitario, creando una subred en el entorno simulado por cada subred existente en el entorno real. La única diferencia reseñable es que en el entorno de Amazon se utiliza una sola red pública, en lugar de las dos existentes en el entorno real, pero este hecho no tiene ninguna implicación a la hora de configurar HTCondor.

Para simular los recursos de los laboratorios se han utilizado instancias *t1.micro*, al ser las más baratas ofrecidas por Amazon, creadas dentro de la subred adecuada. Además, para facilitar el despliegue posterior en el entorno real, se los recursos se han instanciado con el mismo sistema operativo utilizado por los recursos de nuestro departamento (CentOS 6.4 de 64 bits).

Para el despliegue y configuración de HTCondor en el entorno creado se han utilizado diferentes AMIs (*Amazon Machine Images*) que permiten gestionar y replicar de forma sencilla los componentes del sistema. Concretamente, se han creado tantas AMIs como tipos de recursos diferentes hay en el sistema: una para el nodo central de gestión, otra para los recursos de ejecución en redes públicas y otra para los recursos de ejecución en redes privadas.

Finalmente, una vez probado el sistema en Amazon EC2, se ha procedido a la implantación del mismo en el entorno real. Para ello, tan sólo ha sido necesario instalar HTCondor en los recursos del departamento y replicar la configuración usada en Amazon con pequeñas modificaciones en los ficheros de configuración.

4.2 Validación del mediador de recursos efímeros

Para la validación del mediador se ha utilizado el entorno creado en Amazon (figura 5), lanzando una máquina virtual por laboratorio. La validación se ha realizado sobre el entorno simulado y no sobre el real para garantizar que las pruebas se realizaban en las mismas condiciones y eran comparables.

En cuanto a la ocupación de los recursos, se han tomado cuatro horas que consideramos suficientemente representativas de una situación habitual de ocupación. La tabla 1 muestra dicha ocupación indicando para cada hora y laboratorio si los recursos están libres u ocupados. Para simular el horario, los experimentos se han configurado para que el cambio de los recursos de libre a ocupado se produzca 5 minutos después del comienzo de la hora y el paso de ocupado a libre se produzca 5 minutos antes del fin de la hora.

Tabla 1: Horario de ocupación de los laboratorios durante los experimentos

	Lab 0.01	**Lab 0.02**	**Lab 0.03**	**Lab 0.04**	**Lab0.05**	**Lab 1.02**
1ª hora	Ocupado	Libre	Ocupado	Libre	Libre	Libre
2ª hora	Ocupado	Libre	Ocupado	Ocupado	Libre	Ocupado
3ª hora	Ocupado	Ocupado	Libre	Ocupado	Libre	Ocupado
4ª hora	Libre	Ocupado	Libre	Libre	Libre	Ocupado

En cuanto a los experimentos, se ha realizado una prueba de concepto probando dos configuraciones, una con *tareas largas* (50 minutos) y otra con *tareas cortas* (10 minutos), con el fin de valorar la influencia de la duración de las tareas ejecutadas y evaluar la efectividad del mediador de recursos efímeros ante diferentes tipos de tareas. En ambos casos, cada hora se lanza un número de tareas que equivale a un tiempo total de ejecución de 100 minutos.

La tabla 2 muestra los resultados obtenidos en los experimentos. En ella podemos observar el número total de tareas ejecutadas, la duración de cada una, la cantidad de tareas sometidas simultáneamente al comienzo de cada hora y los resultados en cuanto al número de tareas que han sido expulsadas. El número de expulsiones es el número de veces que una tarea ha tenido que ser reubicada al ejecutarse inicialmente en un recurso que posteriormente fue ocupado.

Tabla 2: Resumen de los resultados de la experimentación

Tipo de experimento	**Tareas largas**		**Tareas cortas**	
Nº de tareas	8		40	
Duración	50		10	
Nº de tareas simultáneas	2		10	
Uso de planificador	Sí	No	Sí	No
Nº de expulsiones	0	3	5	5

En el experimento con tareas largas, la utilización del planificador consigue evitar la aparición de expulsiones ya que el éste garantiza que los trabajos se ejecutan en recursos que van a estar libres. En cambio, si no se usa el planificador existe la posibilidad de que se ejecuten trabajos en recursos que posteriormente estarán ocupados ya que la asignación la realiza HTCondor de forma aleatoria.

Sin embargo, en el experimento con tareas cortas se obtienen los mismos resultados con y sin planificador. Esto era lo esperado ya que que el número de tareas a ejecutar cada hora es mayor que el número de recursos libres. De esta forma, aunque las primeras tareas se envían a los recursos libres indicados por el planificador, el resto se envían a recursos que posteriormente pasarán a estar ocupados, y, por tanto, serán expulsadas. Este comportamiento es el esperado ya que el sistema intenta aprovechar el máximo número de recursos ya que el hecho de que un recurso esté reservado no garantiza que vaya a ser ocupado y podría permanecer libre.

Los resultados de esta prueba de concepto nos han permitido validar el funcionamiento del mediador de recursos efímeros y también han permitido extraer algunas conclusiones que nos han permitido establecer líneas futuras de trabajo. La principal conclusión es que el mediador es capaz de recomendar los recursos más adecuados para ejecutar tareas en situaciones en las que se conoce con seguridad la ocupación futura de los recursos. Además, se ha observado la importancia de elegir un tamaño adecuado de tarea.

En general, es deseable ejecutar tareas cortas para aprovechar todos los recursos disponibles y porque en caso de que un recurso pase a ser ocupado, la pérdida de tiempo es potencialmente menor. Sin embargo, la ejecución de tareas más largas en recursos que sabemos que van a estar disponibles puede beneficiarnos al evitar la aparición de fallos. Por tanto, parece que el comportamiento óptimo sería utilizar un número de tareas igual al número de recursos libres en cada momento, si bien queremos estudiar esto en el futuro.

5 Conclusiones y trabajo futuro

En este artículo se han integrado los recursos académicos disponibles en nuestro departamento dentro de un framework de computación, con el fin de utilizarlo como infraestructura de cómputo aplicada a la resolución de problemas complejos. Para ello, se ha desplegado el *middleware* de gestión de recursos HTCondor sobre un entorno simulado en el *cloud* de Amazon. Posteriormente, tras realizar una configuración adecuada del entorno, se ha implantado el sistema en el entorno físico real. El uso de un *cloud* público para desplegar y configurar HTCondor simulando el entorno físico real nos ha aportado varias ventajas en cuanto a la gestión y administración de los recursos y ha permitido disminuir el tiempo necesario para desplegar el sistema en comparación con el tiempo que hubiese sido necesario para llevarlo a cabo en el sistema real.

Para la integración de esta nueva infraestructura en el *framework* de computación previamente desarrollado por los autores se ha extendido uno de sus componentes, incluyendo un planificador que tiene en cuenta el horario de ocupación de los recursos. De esta forma, se puede guiar la asignación de trabajos en diferentes recursos minimizando la expulsión de trabajos en ejecución, mejorando el aprovechamiento de los recursos disponibles y reduciendo el tiempo total de ejecución de problemas computacionalmente costosos.

Como principal línea de trabajo futuro se plantea el estudio de diferentes algoritmos de planificación. Actualmente, el planificador sólo hace uso del horario de ocupación de los recursos, pero también se pretende utilizar el histórico de disponibilidad de los recursos ya que el hecho de que un recurso esté reservado no implica que vaya a ser utilizado y el hecho de que un recurso no esté reservado no implica que esté libre, por lo que la información histórica puede ayudar a localizar y tratar estas situaciones. Finalmente, también hemos observado que una política que puede mejorar el tiempo de ejecución total es ejecutar un número de tareas igual al número de recursos disponibles, de esta forma, en el futuro pensamos investigar la posibilidad de que el mediador sea capaz de agrupar varias tareas de forma dinámica en función del estado de los recursos y la ocupación prevista.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el proyecto JIUZ-2013-TEC-03 y la Red Temática Científico-Tecnológica en Ciencias de los Servicios (TIN2011-15497-E) financiada por el Ministerio de Economía y Competitividad de España. Los autores también quieren agradecer a Marcos Molina su colaboración en la implementación del sistema presentado en este artículo.

Bibliografía

1. Taylor, I.J., Deelman, E., Gannon, D., Shields, M.: Workflows for e-Science. Springer-Verlag London Limited (2007)
2. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
3. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. **39**(1) (2008) 50–55
4. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceedings of the Grid Computing Environments Workshop. GCE '08 (2008) 1–10
5. Kondo, D.: Scheduling task parallel applications for rapid turnaround on desktop grids. PhD thesis, University of California, San Diego (2005)
6. BOINC. `http://boinc.berkeley.edu/` Accedido 25 Abril 2014.
7. Kacsuk, P., Kovacs, J., Farkas, Z., Marosi, A.C., Gombas, G., Balaton, Z.: Sztaki desktop grid (szdg): a flexible and scalable desktop grid system. Journal of Grid Computing **7**(4) (2009) 439–461
8. HTCondor. `http://research.cs.wisc.edu/htcondor/` Accedido 25 Abril 2014.
9. OpenStack Software. `http://www.openstack.org/` Accedido 25 Abril 2014.
10. OpenNebula Software. `http://www.opennebula.org/` Accedido 25 Abril 2014.
11. Eucalyptus Software. `https://www.eucalyptus.com/` Accedido 25 Abril 2014.
12. Fabra, J., Hernández, S., Ezpeleta, J., Álvarez, P.: Solving the interoperability problem by means of a bus. an experience on the integration of grid, cluster and cloud infrastructures. J. Grid Comput. (2013) 1–25
13. Amazon Elastic Compute Cloud (Amazon EC2). `http://aws.amazon.com/ec2/` Accedido 25 Abril 2014.

Towards the user-centric analysis of the availability in IaaS

Antonio Manuel Gutiérrez-Fernández, Pablo Fernández, Manuel Resinas, Antonio Ruiz-Cortés

School of Computer Engineering
University of Seville
{amgutierrez, pablofm, resinas, aruiz}@us.es

Abstract. Availability is a key property in computational services and, therefore, is guaranteed by Service Level Agreements (SLAs) from the majority infrastructure services, such as virtualization (Amazon EC2, Windows Azure, Google Cloud, Joyent, Rackspace, ...) and storage (Amazon S3, Google Cloud Storage, ...). These SLAs describe availability in natural language and there are important differences in the scope and penalties that each service provides. Furthermore, descriptions use specific domain terms so they are difficult to understand by service customers. These circumstances make that availability analysis is a tedious, error-prone and time-consuming task. In this paper, we describe in detail this problem and provide a first approach to deal with these SLAs supported on current SLA analysis techniques.

Keywords: Service Level Agreements, Availability, IaaS, Cloud

1 Introduction

Nowadays, cloud services are massively used to support enterprise software infrastructure. Cloud customers outsource infrastructure services to provide their own services. Service Level Agreeemnts (SLAs) guarantee the service consumption between the different parties. Cloud customers act as service providers for third parties. A SLA for typical software provision guarantees availability periods (24x7, office time, ...) or performance (requests per second, request latency,...) and the service provider responsability on these guarantees. As external service provision bases on cloud services, customers examine cloud services SLA so their responsability relates to the infrastructure guarantee. However, main cloud providers, such Amazon, Google, Rackspace or Joyent provide a non-negotiable Service Level Agreements (SLAs) where guarantee terms semantic is far away from software service provision semantics.

These SLAs are described in a natural language so they are easily understood by customers. SLAs include terms which provider guarantees and penalties policies. These terms are mainly constraints over some quality conditions such a availability, latency or performance. The variables constrained depend on service type (computation, storage, network, database, etc).

Service availability is very important for customers and all cloud service providers offer guarantees related to availability. In this paper, we focus on analysis of guarantees over availability domain from the customer perspective. To plan infrastructure deployment, customers evaluates how the IaaSS providers gurantees match their own requirements. In spite of natural language guarantees are easily understable, manually evaluating how these guarantees apply to customer requirements is a time consuming, tedious and error-prone task, so automating this analysis has an impact on customers business plan [1].

The main infrastructure services are computing and storage resources so we focus in both kind of services. Particularly, we take SLAs where single machine availability is guaranteed. Each kind of services have different semantic about availability and customer preferences are analysed considering these differences. To afford this analysis, we model usual customer preferences in the form of Frequently Answered Questions (FAQ). First step to answer these questions, is translate natural language description to a formal language that can be computationally operated. WS-Agreement[1] is a well-known and widely used schema to describe Agreement supporting penalties and rewards terms so we used to support our approach. In a second step, as there is not existing tool or solution which automate the availability checking and penalties appliance operations, we design and develop these operations to automate answering the proposal questions.

We introduce three basic questions of interest to analyse providers guarantee for infrastructure services. These questions are:

- Q1: Given Availability guarantee, which is the maximum down time without penalties?
- Q2: Which is the penalty when a machine has been down for N minutes (in a row or in separate periods)?
- Q3: Which is the minimun time with the maximum penalty (when the guarantee limit is reached)?

In next sections, SLAs from cloud providers are described for computing (2) and storage services (3). Section 4 identifies research efforts to model SLA supporting penalties definition in WS-Agreement and design automate solutions for these questions as analysis operations.

2 Availability in computing services

2.1 Rackspace

In the Rackspace SLA, host availability is defined with the following terms:

We guaranty the functioning of all cloud server hosts including compute, storage, and hypervisor. If a cloud server host fails, we guaranty that restoration or repair will be complete within one hour of problem identification. If we fail to meet a guaranty stated above, you will be eligible for a credit. Credits will be

[1] `http://www.ogf.org/documents/GFD.107.pdf`

calculated as a percentage of the fees for those Cloud Servers adversely affected by the failure for the current monthly billing period during which the failure occurred (to be applied at the end of the billing cycle), as follows: Cloud Server Hosts: 5% of the cloud server fees for each additional hour of downtime, up to 100% of the cloud server fees

According to the SLA defined, the Rackspace availability per machine guarantee excludes the first 60 minutes and offers a 5% per every 60 minutes over the first ones. So the questions proposed can humanly be answered.

- Q1: Maximum down time without penalties is 61 minutes.
- Q2: After N down minutes in a row,
  the penalty is 0 if $N \leq 60$minutes, 5% of monthly fee if $60 < N \leq 120$
  10% of monthly fee if $120 < N \leq 180$ ... 100% of monthly fee if $N > 1260$
- Q2: After N down minutes in separate periods, the penalty is calculated per separate period.
- Q3: As in previous question, answer depends on the distribution of offline periods. Having a distribution of N offline periods, and a 5% of penalty per 60 minutes, maximum penalty, 100%, is reached when the sum of minutes (minus 60) of every offline period is 20×60 minutes (i.e.: 1200 minutes). In the simple case, a single offline period, maximum penalty is reached in 1260 minutes.

2.2 Joyent

In the Joyent SLA, host availability is defined with the following terms:

Goal: Joyents goal is to achieve 100% Availability for all customers. Remedy: Subject to exceptions, if the Availability of customers Services is less than 100%, Joyent will credit the customer 5% of the monthly fee for each 30 minutes of downtime (up to 100% of customers monthly fee for the affected server).

So, Joyent guarantees any unavailable machine time and the penalty is 5% per each 30 unavailable minutes. The answer to proposal questions is similar to Rackspace

- Q1: There is no downtime without penalties.
- Q2: Penalty is 5% of monthly time if $0 < N <= 30$ minutes, 10% of monthly fee if $30 < N \leq 60$ ... 100% of monthly fee if 300 ¡ N
- Q3: In this case, maximum penalty is 100% and as penalty is 10% per 30 minutes, maximum is reached in 10×30 minutes (i.e.: 300 minutes).

3 Availability in storage services

Storage services are usually binded to computing services but as operative nature is different from computing, availability guarantees are described with different semantics.

As example scenarios for these questions, we take Google Cloud and Amazon S3 storage services.

3.1 Amazon S3

In the Amazon S3, storage availability guarantee is defined with two key concepts:

- Error Rate: Number of error requests divided by total number of requests in a 5 minute time period.
- Monthly Uptime Percentage (MUP): 100% minus the average of error rates in a month.

This is, monthly uptime percentage depends on the request distribution per 5 minutes periods and the failure rate. Possible penalties depends on the MUP with following rule:

- MUP is greater or equal than 99% but less than 99.9%, penalty is 10% of the credit for next payment.
- MUP is lesser than 99%, penalty is 25% of the credit for next payment.

Analysing this SLA, the answer to provided questions are:

- Q1: Maximum time without penalties depends on request distribution. Assuming 100% failure and at least one request per 5-minute period, there wouldn't be any penalty until 9 5-minute intervals (0.1% percent of monthly 5-minutes periods). So, answer to question is that maximum period without penalties is 45 minutes (9×5), the specific requests number depends on request ratio. For example, with 1 request per minute ratio, with 45 failed requests customer could apply for penalty or with 60 requests per minute ratio, customer would have 2700 failed requets before apply penalties (i.e.: considering a distribution of A requests per minute, the maximun failing requests is $9 \times 5 \times A$ requests).
- Q2: For this question, we can consider two cases. In one hand, a distribution of consecutive 100% failure requests. In this case, the penalty depends entirely of the time period considered. Over 45 minutes, customer gets a 10% of service credit and over 450 a 25%. In the other hand, if failure requests are not in a row, penalty applied depends on the distribution of failure requests through time.
- Q3: Similarly to first question, over 1% percent of the time periods with 100% failed requests (i.e.: over 450 minutes), customers get maximum credit for next billings (i.e.: 25%). For a failure regular distribution of C failures per requests (C has to be lesser or equals than 1), the maximum penalty is applied when $0.01 \times MonthlyMinutes \div C$ minutes have passed.

3.2 Google Cloud Storage

Google cloud storage SLA describe its availability guarantee, similarly to S3, around the request error rate. In this case, key concepts are:

– Error Rate: Number of error requests divided by total number of valid requests.
– Downtime Period: 10 consecutive minutes interval where Error Rate is bigger than 5% percent.
– Monthly Uptime Percentage (MUP): Total minutes in a month minus the number of the minutes in downtime periods divided by total minutes in a month.

Google offers two different SLAs to guarantee storage availability. We consider standard SLA as the concepts are similar in both of them and differences are bigger availability at bigger price. Standard SLA offers following penalty:

– MUP is greater or equal than 99% but less than 99.9%, penalty is 10% of the credit for next payment.
– MUP is greater or equal than 95% but less than 99%, penalty is 25% of the credit for next payment.
– MUP is lesser than 95%, but penalty is 50% of the credit for next payment.

So, similar to Amazon, the answers to the proposed questions for Google guarantees depend on the request distribution:

– Q1: Similarly to Amazon S3, considering 100% failure and at least one request per 10-minute period, until minute 50 (0.1% percent of monthly 10-minutes periods), there wouldn't be any penalty. The maximun number requests depend on request ratio on those 50 minutes. Considering a distribution of A requests/minute, the maximun failing requests is $50 \times A$ requests.
– Q2: To answer this question, again, as in Amazon S3, we can differentiate between the simple case, i.e.: a distribution of consecutive 100% failed requests (penalty depends exclusively on time) or a non regular failed requests distribution, where penalty depends on this distribution.
– Q3: In this question, although penalty is calculated in a similar way to Amazon S3, Google offers higher guarantees for pesimistic scenarios. If time intervals with failures get a 5% (i.e.: over 2160 minutes with 100% failed requests or, with a C failure distribution, $0.01 \times MonthlyMinutes \div Cminutes$) customer gets a 50% service credit.

4 Our Proposal

As far as we know, existing solutions focus on cloud monitoring or validate SLAs but no one focuses on the analysis at design time of resource availability guaranteed by cloud providers.

First step towards supporting automate analysis over availability is modelling the different guarantees proposed so that their penalties can be checked by a software component. Our proposal is based on WS-Agreement which is a standard that is widely used and that has been successfully applied in the computational domain. WS-Agreement is the prominent proposal to model SLAs and a number

of supporting tools for editing and analysing WS-Agreement documents, such our agreement management environment IDEAS[2].

4.1 Modelling SLA with WS-Agreement

WS-Agreement specification defines an Agreement document metamodel. This metamodel is composed of several distinct parts: An optional name, context and terms. The context section provides information about participants in the agreement (i.e., service provider and consumer) and agreement's lifetime. Terms section describe the agreement itself. There are two different kind of terms, namely service terms and guarantee terms, that can be recursively composed.

Guarantee Terms define constraints over Service Properties. They constrain the service execution that an obligated party must fulfill. These constraints are referred as Service Level Objectives (SLOs). So an SLO is an assertion over monitorable properties defined in Service Propeties. Guarantee Terms definition can be guarded by a Qualifying Condition (QC), which indicates a precondition to apply the constraint in the SLO. The valuation of the guarantee is defined using Business Value List ([2],[4]). Business Value includes the expression of importance to determine the possible penalties in a determined interval. Expressions to define SLOs and Penalties are used to operate over Agreements (i.e.: check an SLO is violated to renegotiate SLA or calculate penalties).

Using WS-Agreement document structure, natural language SLAs provided by Joyent and Rackspace are defined in a straight way (Figures 1 and 2). Syntax used in figures is iAgree, a human readable syntax proposed for WS-Agreement documents that provided a specific language to define SLO expressions. iAgree is a proposal from [3], which includes mapping from iAgree to WS-Agreement and the opposite one. Original iAgree proposal does not describe Business Value List sections so we extend it using same expression language to define Penalty expressions.

As figures depict, the Availability guaranteed by Rackspace and Joyent correspond to Service Level Objective of Unavailability lower than 60 minutes or equal to zero minutes, respectively. To simplify scenario, guarantee limitations (as planned maintenance) is not defined but it could be included as qualifying conditions for the guarantee terms. Penalties are described in Business Values for the monthly payment interval using a math expression according to the natural language definition.

4.2 Automate availability analysis

The analysis of agreements means extracting relevant information from these documents so it is often useful to define analysis process as operations that take a set of parameters as input, and return a result as output [3].

With the SLA provided in previous section and the iAgree modelling proposed (Figures 3, 4, 5, 6), we analyse how to define operations that answer to

[2] `http://www.isa.us.es/IDEAS`

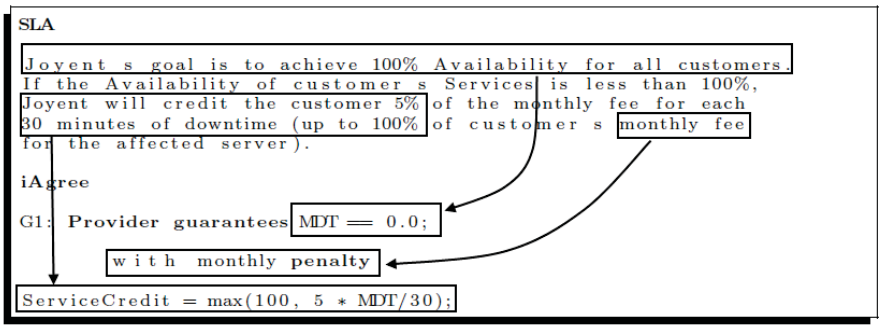

Fig. 1. Modelling Joyent SLA availability guarantee in iAgree

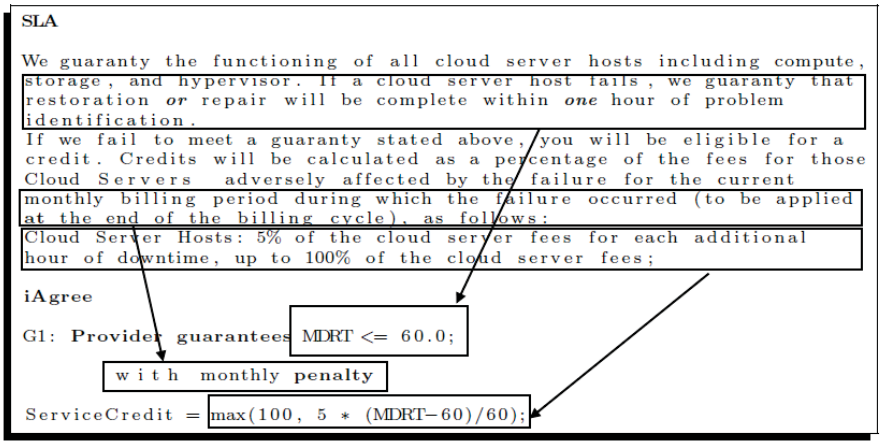

Fig. 2. Modelling Rackspace SLA availability guarantee in iAgree

provided questions. In spite of these operations being analysed for the example providers, we describe them for any cloud provider SLA. As SLO expressions in iAgree are defined as constraints over service properties, operating with such expression can be afford with a logic solver, such as Constraint Satisfaction Problems (CSPs). So we express the proposed questions as analysis operations.

4.3 Maximum failure without penalties operation

Considering the provided scenario, given availability guarantee terms, the operation to answer the question about maximum failure interval returns a time interval. This interval can be expressed as time interval or number of requests depending on service type.

To implement these operation, we consider that guarantee term expression is equivalent to penalty expression (i.e.: whenever an SLO is violated, a penalty is applied and viceversa). So the answer to this operation does not need evaluate penalty expression and compensation and can be checked with only Service Level Objective expression (SLO).

4.4 Applied Penalties operation

Penalty analysis depends on the failure rate. Considering the unavailability Service property (MDT in Joyent Agreement, MDRT in Rackspace Agreement or MUP in Amazon S3 and Google Storage), we get the result of this operation.

Given time input variable, constraint solution should return an expression over Penalty Value Unit. Unlike previous operation, solution to this operation depends on the availability guarantee is expressed as accumulate value or not accumulate one (i.e.: Joyent offers guarantee over any unavailable time but Rackspace only for exceding time over 60 minutes per service interruption). So, considering the simple case, this is, where failure ratio is 100% per unavailable interval, both solutions are:

- Accumulated value: Solution is calculated using the sum of every unavailable interval length.
- Non-accumulated value: Solution is the sum of solution for every single unavailable interval length

4.5 Minimum time with maximum penalties operation

Considering the provided scenario, given availability guarantee terms, the expected answer to the question is the minimum time period where maximum possible penalty are appliable.

To implement these operation, we consider that penalty expression and solve for maximum penalties. For instance, in Joyent solving the single penalty expression:

$$P = min(100, 5 \times T \div 30)$$

```
Template RackspaceSLA version 1
Provider Rackspace as Responder;
...
Guarantee Terms
  G1: Provider guarantees MRT <= 60.0
    with monthly penalty
    ServiceCredit = 5 * (MRT - 60) / 60
...
```

Fig. 3. Computing SLA provided by Rackspace *iAgree*

```
Template JoyentSLA version 1
Provider Joyent as Responder;
...
Guarantee Terms
  G1: Provider guarantees MDT == 0.0
    with monthly penalty
    of ServiceCredit = 5 * MDT / 30
...
```

Fig. 4. Computing SLA provided by Joyent *iAgree*

```
Template AmazonS3SLA version 1
Provider Amazon as Responder;
...
Guarantee Terms
  G1: Provider guarantees MUP <= 99.9%
    with monthly penalty
    of ServiceCredit = 10 if MUP >= 99% AND MUP < 99.9%
    of ServiceCredit = 25 if MUP < 99%
...
```

Fig. 5. Storage SLA provided by Amazon S3 *iAgree*

```
Template GoogleCloudSLA version 1
Provider Google as Responder;
...
Guarantee Terms
  G1: Provider guarantees MUP <= 99.9%
    with monthly penalty
    of ServiceCredit = 10 if MUP >= 99% && MUP < 99.9%
    of ServiceCredit = 25 if MUP >= 95% && MUP < 99%
    of ServiceCredit = 50 if MUP < 95%
...
```

Fig. 6. Storage SLA provided by Google Cloud *iAgree*

Being maximum Penalty = 100%,

$$MinimunTime = 100 \times 30 \div 5 = 20 \times 30 = 600 minutes.$$

In Amazon S3, similarly, Being Maximun Penalty = 25%, it is reached when unavailability = 1%. With a 100% failed requests ratio: 1% is 86,4 5-minutes intervals or 432 minutes. For lower failed requests ratio, time to get maximun penalty would be inverse to the failure ratio:

1 / 3 failed request ratio would take $3 \times 86,4$ intervals to reach maximun penalty.

5 Conclusions and Future Work

The contribution of this paper is based on automating availability analysis procedure which can be applied to any provider with availability guarantees. Availability guarantees are not expressed in a homogenous semantic in the different infrastructure providers. Guarante Terms refered to availability usually express penalties. These penalties easily reflects different proposals from the provider goals, so it is advisable extend common validation criteria in SLA and its analysers and compilers to detect related errors.

As questions about availability are not usually straight to answer and, in all cases, are tedious and error-prone, automating these questions support terms analysis. Expressing these questions as analysis operations over SLA simplifies infrastructure solutions design and fasts the development of concept proof and time to market.

This proposal focuses on analyse availability SLA guarantees in major infrastructure providers. However, early analysis detects availability comprises a wide range of semantics depending on the cloud provider and kind of services. Therefore, this work can be extended to define validation criteria in providers where availability concept has a wider scope and greater complexity or different domains such as Platform as a Service (PaaS) or Software as a Service (SaaS). Furthermore, analysis operations over availability are designed to check SLAs guarantees at service design and planning stage but it is not reviewed how this operations can apply to execution and monitoring phases.

References

1. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: Sybl: An extensible language for controlling elasticity in cloud applications. In: CCGRID. pp. 112–119 (2013)
2. Ludwig, H., Ludwig, H.: Ws-agreement concepts and use agreement-based service-oriented architectures. Tech. rep. (2006)
3. Müller, C.: On the Automated Analysis of WS-Agreement Documents. Applications to the Processes of Creating and Monitoring Agreements. International dissertation, Universidad de Sevilla (2013)
4. Rana, O., Warnier, M., Quillinan, T., Brazier, F., Cojocarasu, D.: Managing violations in service level agreements. In: Grid Middleware and Services, pp. 349–358. Springer US (2008), http://dx.doi.org/10.1007/978-0-387-78446-5_23

Priority-Based Human Resource Allocation in Business Processes*

Cristina Cabanillas[1], José María García[2], Manuel Resinas[3], David Ruiz[3], Jan Mendling[1], and Antonio Ruiz-Cortés[3]

[1]Vienna University of Economics and Business, Austria
{cristina.cabanillas,jan.mendling}@wu.ac.at
[2] STI Innsbruck, University of Innsbruck, Austria jose.garcia@sti2.at
[3]University of Seville, Spain {resinas,druiz,aruiz}@us.es

Summary of the Contribution

Business Process Management Systems (BPMS) are increasingly used to support service composition, typically working with executable BP models that involve resources, which include both automatic services and services provided by human resources. The appropriate selection of human resources is critical, as factors such as workload or skills have an impact on work performance. While priorities for automatic services are intensively researched, human resource prioritization has been hardly discussed. In classical workflow management, only resource assignment at BP design time to select potential performers for activities, and resource allocation at run time to choose actual performers, are considered. There is no explicit consideration of prioritizing potential performers to facilitate the selection of actual performers. It is also disregarded in professional solutions.

In this paper, we address this research gap and provide two contributions: (i) we conceptually define prioritized allocation based on preferences; and (ii) we propose a concrete way in which preferences over resources can be defined so that a resource priority ranking can be automatically generated. Our solution builds on the adaptation of a user preference model developed for the discovery and ranking of semantic web services called SOUP [1] to the domain at hand. As a proof of concept, we have extended the resource management tool CRISTAL (`http://www.isa.us.es/cristal`) with the SOUP component [2], using RAL [3] for resource selection.

1. J. M. García, D. Ruiz, and A. R. Cortés, "A Model of User Preferences for Semantic Services Discovery and Ranking," in *ESWC (2)*, pp. 1–14, Springer, 2010.
2. J. M. García, M. Junghans, D. Ruiz, S. Agarwal, and A. R. Cortés, "Integrating semantic Web services ranking mechanisms using a common preference model," *Knowl.-Based Syst.*, vol. 49, pp. 22–36, 2013.
3. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Defining and Analysing Resource Assignments in Business Processes with RAL," in *ICSOC*, vol. 7084, pp. 477–486, Springer, 2011.

* This work was published in ICSOC 2013, vol. 8274, 374-388. It was partially supported by the EU-FP7, the EU Commission, the Spanish and the Andalusian R&D&I programmes (grants 318275, 284860, TIN2009-07366, TIN2012-32273, TIC-5906).

Una solución para la gestión e integración de Internet de las Cosas en la Nube

Adrián Nieto, Javier Cubo, y Ernesto Pimentel

Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación, España
{adrian,cubo,ernesto}@lcc.uma.es

Resumen. La falta de estandarización a la hora de conectar dispositivos a Internet del Futuro origina un problema relativamente novedoso en el que aún no se ha definido una línea concreta de actuación. En este sentido, el estándar OASIS DPWS (*Device Profile for Web Services*) está orientado a exponer, con independencia de su capacidad, dispositivos de forma genérica basándose en la pila de protocolos para Servicios Web. Sin embargo, el alcance de la comunicación con dichos dispositivos de forma ubicua se encuentra limitado al uso de *discovery proxies*, que agregan dispositivos conectados a diferentes redes. Esto dificulta enormemente la vision global de dichos dispositivos, además de delegar en primera instancia todas las tareas de comunicación a un único punto de entrada. En este trabajo, se propone extender el estándar DPWS para permitir la creación de un 'repositorio' de dispositivos en la Nube, donde considerando los beneficios de la computación en la Nube, como su capacidad 'ilimitada', se almacenen, procesen y orquesten la gran cantidad de dispositivos que constituyen las nuevas aplicaciones de la sociedad de Internet del Futuro.

Palabras clave: Internet de las Cosas, Internet del Futuro, Orquestación, Computación en la Nube, Nube de las Cosas, Dispositivos heterogéneos, Plataforma en la Nube, Dispositivo como Servicio

1. Introducción

Los avances en el uso de dispositivos de capacidad limitada y el auge de tecnologías de comunicación de bajo consumo como LR-WPAN (*Low-Rate Wireless Personal Area Network*) o RFID (*Radio Frequency IDentification*), están permitiendo cada vez mas otorgar a objetos cotidianos la categoría de 'objetos inteligentes'. Como parte de Internet del Futuro, Internet de las Cosas (*Internet of Things, IoT*) hace referencia a un paradigma donde dispositivos y objetos de la vida cotidiana (dispositivos móviles, sensores, cámaras, impresoras, electrodomésticos, etc.) están conectados entre sí. La idea supone un reto al que enfrentarse y a su vez, persigue el objetivo de mejorar la sociedad en su actividad diaria [4].

Las aplicaciones pertenecientes a Internet del Futuro pueden modelarse haciendo uso de SOA (*Service-Oriented Architecture*) [6]. De esta manera, es posible garantizar la interoperabilidad entre todos los objetos partícipes en mundo físico y virtual, interactuando de forma dinámica entre sí. Enfoques recientes se

han centrado en aplicar SOA directamente sobre dispositivos [12]. Sin embargo, integrar esta arquitectura sobre un dispositivo no es un problema trivial.

Para abordar esta cuestión DPWS (*Devices Profile for Web Services*) [1], un estándar de OASIS, proporciona la funcionalidad requerida pudiendo llegar a considerarse como el puerto USB a través de internet. Tiene objetivos similares al ya implantado UPnP (*Universal Plug and Play*) pero con una visión más centrada en especificar cada dispositivo como un servicio.

Además, con el auge de las plataformas *cloud* son cada vez más necesarias soluciones completas que proporcionen una conectividad heterogénea [5], donde la información resida y se procese fuera de nuestros dispositivos locales, de cara a favorecer la interacción entre entes, sin importar las características propias de los objetos que actúan en el sistema. Esta visión de los dispositivos como servicios (*Thing as a Service*, TaaS) es trascendental para la propuesta que presentamos, ya que una vez especificada la forma de comunicación con los objetos, es factible imaginar una capa abstracta por encima de estos, que actúe como lo hace un sistema gestor de contenidos (CMS, *Context Management System*) en el caso de una página web; permitiendo que la interacción con los objetos sea, tanto para el desarrollador como para el usuario final, una tarea intuitiva.

La principal diferencia de la Nube frente a las alternativas tradicionales es la capacidad de adaptarse automáticamente a las diferentes cargas de trabajo sin que el usuario final o el desarrollador lo perciban. Esta característica distintiva se denomina elasticidad y es, junto con la replicación automática y prevención de fallos, una de las cualidades más importantes de los entornos *cloud*.

El resto de este artículo está organizado según la siguiente estructura: La Sección 2 presenta una breve introducción del estándar DPWS. La Sección 3 describe nuestra propuesta de integración en la Nube de la visión TaaS, comparando en la Sección 4 otras alternativas disponibles. Finalmente, presentamos nuestras conclusiones en la Sección 5 .

2. Breve introducción a DPWS

DPWS define una de conceptos sobre la pila de protocolos WS-* enfocados a dispositivos con recursos limitados. Al estar basado en WS-*, facilita la coexistencia entre los dispositivos y otros servicios web ya existentes. En contraposición a la utilización de WS-* como estándar para IoT, algunos autores proponen enfoques basados en REST [7]. Sin embargo, y aunque los servicios web RESTful son más sencillos, a la hora de lidiar con funcionalidades complejas como comportamiento, semántica o calidad de servicio resultan insuficientes [8].

La especificación de DPWS fue publicada en Mayo de 2004. La primera versión funcional de DPWS es la 1.1, aprobada como estándar OASIS junto con WS-Discovery y SOAP-over-UDP en Junio de 2009 [1]. DPWS define dos tipos de servicios: aquellos relacionados con la funcionalidad ofrecida y servicios de alojamiento a otros servicios. Los dispositivos están directamente relacionados con el servicio de alojamiento, el cual juega un papel clave en el proceso de descubrimiento, ya que sirve de capa externa de acceso a la funcionalidad que

proporcionan los servicios intrínsecos de los dispositivos, los cuales son, a grandes rasgos, simplemente operaciones WS-* tradicionales.

El dispositivo DPWS puede funcionar de dos maneras distintas. La primera se denomina modo *ad-hoc*, en el cual las búsquedas y los mensajes globales se envían y reciben por *multicast* dentro de las subred correspondiente. La segunda conforma la posibilidad de crear intermediarios, denominados *discovery proxy*, para solventar las limitaciones impuestas por el modo de *ad-hoc*:

- La necesidad de permanecer en el mismo rango de *multicast* que el dispositivo que buscamos.
- Reducir el tráfico residual de la red generado por las peticiones *multicast*.

Una vez realizado el primer intercambio de metadatos, y conocida la definición del dispositivo y de los servicios alojados mediante el fichero WSDL, se emplean mensajes SOAP para realizar las llamadas a los métodos expuestos, como si de un servicio web tradicional (basado en WS-*) se tratase.

3. Propuesta de integración

La presencia de un *discovery proxy* puede hacer entender al lector que la necesidad de un agregador de dispositivos ha sido resuelta. Sin embargo, DPWS únicamente proporciona la descripción del servicio según su fichero WSDL, sin realizar procesado de ningún tipo sobre la misma. Por otra parte el concentrar todas las peticiones en un *discovery proxy* centralizado (DPWS no define mecanismos de adaptación a la carga como en la Nube) se limita la escalabilidad del sistema y perjudica el rendimiento.

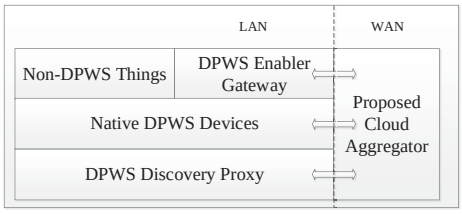

Fig. 1. Agregador propuesto como extensión a DPWS.

Como se muestra en la Fig. 1, nuestra propuesta plantea añadir un nivel más a la actual arquitectura DPWS que permita la posibilidad de utilizar un servicio en la Nube donde 'alojar' los dispositivos, para realizar tareas más complejas como orquestación y análisis de datos, las cuales hemos comenzado a realizar en trabajos previos [2] y pretendemos seguir avanzando con el presente trabajo.

3.1. Definición de los dispositivos en la nube

Las modificaciones realizadas en los archivos WSDL alojados en la Nube se concentrarán en el apartado de documentación previsto en el estándar DPWS, evitando así modificar la definición del estándar.

Nuestra contribución a la definición de los dispositivos tiene tres partes diferenciadas:

- **Identificador de dispositivo único:** Cada definición de un dispositivo (a través de un fichero WSDL) recibirá un identificador único generado por la plataforma inmediatamente después de añadirlo al repositorio, de manera que se pueda hacer referencia de forma sencilla a ese dispositivo concreto.
- **Información de orquestación:** Tomando como base la extensión de comportamiento desarrollada en otros trabajos por algunos autores de la presente propuesta [3], podremos establecer restricciones en el uso de dispositivos compuestos a través de la Nube, gracias a los identificadores únicos antes mencionados.
- **Configuración de la recolección de datos:** En dispositivos de bajo consumo es vital mantener un estado de reposo el máximo tiempo posible. Por ello, es más razonable que sea la propia plataforma quien despierte al dispositivo cuando necesite información de él. La entrada de las operaciones no es necesariamente estática, pues a su vez se permite la composición de servicios de una manera similar a la realizada por *Yahoo Pipes* [9], siendo comprobada la información de orquestación automáticamente por la plataforma para evitar estados no deseados.

Cuadro 1.1. Contribución al estándar

```
<documentation>
...
  <cloudid identifier="Device A" />
  <datacollector>
    <operation name="opA3">
      <interval>5000</interval>
      <input>
        <parameter name="paramA">
          <value>42</value>
        </parameter>
        <parameter name="paramB">
          <value>
            <cloudid identifier="Device B">
              <operation name="opB1" />
              <parameter name="...">
                ...
              </parameter>
            </cloudid>
          <value>
        </parameter>
      </input>
    </operation>
  </datacollector>
  <behaviour>
    <constraint name="C1" type="afterAll">
      <after>
```

```
27          <operation name="opA1"/>
28          <operation name="opA2"/>
29        </after>
30        <before>
31          <operation name="opA3"/>
32        </before>
33      </constraint>
34      <constraint name="C2" type="afterSome">
35        <cloudid identifier="Device B">
36          <after>
37            <operation name="opB1"/>
38          </after>
39        </cloudid>
40        <before>
41          <operation name="opA1"/>
42        </before>
43      </constraint>
44    ...
45    </behaviour>
46  ...
47  </documentation>
```

En el ejemplo del Cuadro 1.1 se muestra la intervención de dos dispositivos, *Device A* cuyas operaciones se denominan *opA1, opA2 y opA3* y un *Device B* con el método *opB1*. Suponemos para el ejemplo que la plataforma dónde se alojan los dispositivos conoce de antemano esta información, ya que al registrar el dispositivo en ella se envió el WSDL original que contenía la definición del mismo.

Las reglas C_i presentadas en el Cuadro 1.1, son las presentadas en el trabajo [3], donde se proponen tres tipos de restricciones que actuarán sobre un conjunto de operaciones:

- $\{b_1, \ldots, b_n\} afterAll \{a_1, \ldots, a_n\}$: Las operaciones $\{b_1, \ldots, b_n\}$ solo podrán ejecutarse después de realizar todas las operaciones $\{a_1, \ldots, a_n\}$.
- $\{b_1, \ldots, b_n\} afterSome \{a_1, \ldots, a_n\}$: Basta con ejecutar una operación $\{a_1, \ldots, a_n\}$ para poder ejecutar cualquiera de las operaciones $\{b_1, \ldots, b_n\}$.
- $onlyOneOf\{a_1, \ldots, a_n\}$: En cada interacción únicamente podrá llamarse a una de las operaciones $\{a_1, \ldots, a_n\}$.

En el caso particular de nuestro ejemplo, sobre *Device A* se han establecido dos reglas de uso, la primera *C1* proviene del WSDL original, las cuales han sido definidas por el fabricante. La segunda, *C2* establece una orquestación entre A y B, pues indica que antes de poder ejecutar *opA1* tendremos que ejecutar *opB1* en *Device B*.

Por último, para poder realizar un análisis de datos, el administrador ha configurado *Device A* de forma que llame a la operación *opA1* cada 5000 milisegundos tomando dos parámetros de entrada, uno estático (*paramA*) y otro que proviene de ejecutar *opB1* en *Device B*. En la recolección de datos, la propia

plataforma es la encargada de establecer automáticamente el orden de ejecución de las operaciones gracias a la información de orquestación proporcionada por todos los ficheros WSDL almacenados en el repositorio.

Esta información de recolección y orquestación se introduce en el portal de la plataforma mediante el uso de *mash-ups*, permitiendo al usuario interactuar de forma sencilla y rápida. Además, gracias a la división en capas (ver la Fig. 1), al permanecer ajenos los dispositivos en las capas superiores es posible definir dos niveles de orquestación, uno para los que acceden directamente a través de DPWS y otro para los accesos a través de la Nube.

3.2. Comunicación con la plataforma en la Nube

La comunicación con la plataforma en la Nube se realizará en los estándares soportados por DPWS. Esta decisión no es determinante para el correcto funcionamiento de la plataforma, de modo que para utilizar otro protocolo (como por ejemplo, JSON-RPC), pero nos evita realizar una traducción entre los mensajes que se envían a los dispositivos mediante DPWS directamente y los que se envían a la nube.

Para llevar a cabo tal tarea, nuestro servicio *cloud* debe disponer de una serie de operaciones generales:

- **Registrar dispositivo:** Recibe un archivo WSDL con la definición del dispositivo cuyos *endpoints* son accesibles desde internet. La operación almacena el dispositivo y genera un identificador único que es añadido a la definición que dispondrá el servidor. En la respuesta comunicará el identificador al cliente.
- **Eliminar dispositivo:** La operación recibe un identificador de dispositivo y elimina todas las referencias a él del sistema.
- **Ejecutar operación:** El cliente envía el identificador del dispositivo junto con el nombre de la operación y los parámetros definidos en el WSDL como lo haría con un dispositivo DPWS normal. El servidor verifica las reglas establecidas en el dispositivo y devuelve el resultado de la operación.
- **Obtener histórico de datos:** El cliente envía el identificador del dispositivo junto con el nombre de la operación y recibe los datos (si existen) almacenados.
- **Añadir comportamiento:** Recibe un identificador de dispositivo, así como un bloque 'behaviour' con las nuevas restricciones.
- **Programar recolección de datos:** La operación necesita un identificador de dispositivo junto con un bloque 'datacollector', actualizando la programación anterior con los nuevos datos.
- **Obtener lista de dispositivos:** Lista los dispositivos registrados en el sistema.

3.3. Casos de uso

Al situarse como una capa adicional permite la interacción mediante DPWS o a través de la Nube de forma independiente. Planteamos tres casos de uso para ejemplificar nuestra propuesta.

Registro de un nuevo dispositivo y acceso local La Fig. 2 representa el proceso que sigue la conexión de un nuevo elemento, el cual carece de conectividad nativa DPWS (como ocurre en el caso de una mota dentro de una red de sensores) por lo que requiere de un *gateway* intermedio que traduzca del estándar de comunicación propietario del dispositivo a DPWS. Asimismo observamos como la Nube únicamente interviene para almacenar un nuevo dispositivo, permitiendo la interacción con él a través de DPWS.

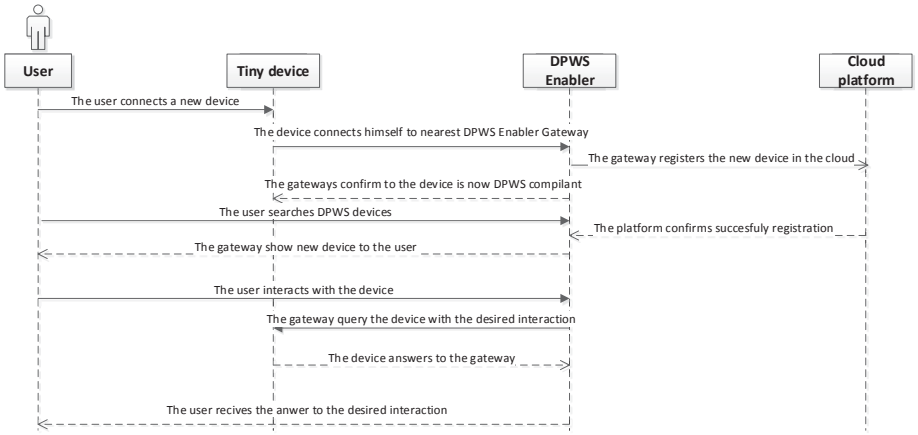

Fig. 2. Registro y interacción dentro del segmento de red accesible (generalmente LAN) mediante DPWS.

Interacción con un dispositivo en la nube El proceso de interactuar con un dispositivo alojado anteriormente en la Nube se ilustra en la Fig. 3. Dicha interacción consiste en consultar el último valor, donde no se modifica el estado ni la relación con la Nube para simplificar el diagrama de secuencia.

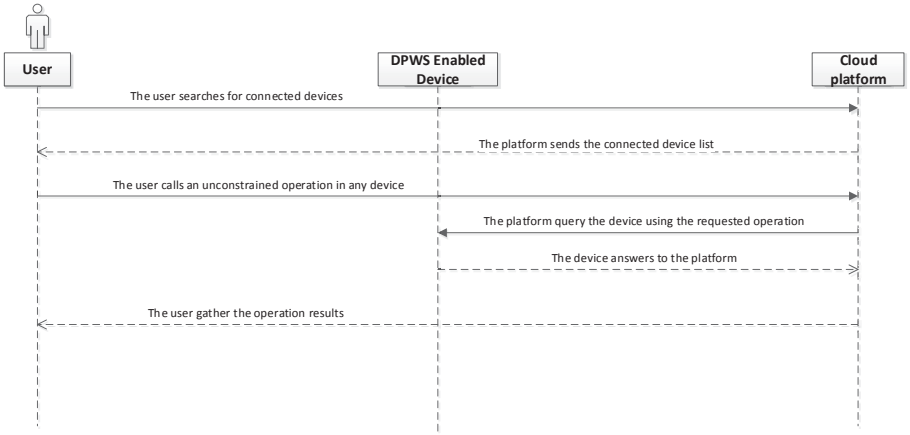

Fig. 3. Obteniendo datos de un dispositivo previamente almacenado.

Programación de una recolección de datos Por último, un caso interesante de estudio es el representado por la Fig. 4 donde hemos tomado como configuración de obtención de datos la aportada en el Cuadro 1.1.

Observamos que aunque explícitamente no se haya mencionado es la propia plataforma la que ha realizado las llamadas a los métodos necesarios para verificar la orquestación.

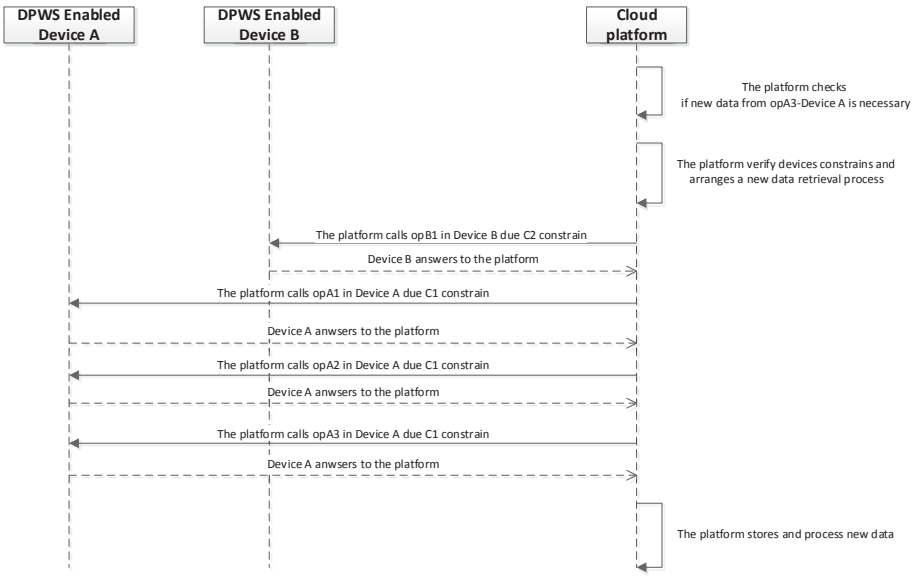

Fig. 4. Proceso automático de adquisición de datos.

En el ejemplo se presupone que las operaciones *opA1* y *opA2* no tienen parámetros de entrada, por lo que la propia plataforma puede hacer la llamada sin intervención del usuario. De no ser así, al proponer la orquestación la plataforma daría un error e informaría al usuario de que existen operaciones que no pueden ser llamadas automáticamente.

4. Trabajo relacionado

La idea de fusionar Dispositivos como Servicio y la Nube representa la punta de lanza de las investigaciones actuales en Internet de las Cosas [11]. Sin embargo y aunque la mayoría de las propuestas convergen en la idea de heterogeneidad, muchas de ellas se encuentran enfocadas a entornos concretos, en su mayoría relacionados con la sanidad o la domótica [5, 10].

Nuestra idea se plantea como una plataforma independiente a las soluciones software comerciales como Xively, Carriots, en la cual podamos realizar las modificaciones necesarias para adaptarla a nuestra línea de investigación. Existe

una iniciativa reciente europea denominada FI-WARE[1] la cual propone entre muchas otras ideas, el uso de plataformas *cloud* de cara al despliegue de redes de sensores. Sin embargo, existen varias diferencias principales con nuestro planteamiento:

- **Estándares empleados:** Emplean REST para la comunicación con la plataforma, el cual es insuficiente para nuestro propósito. Para la definición de los sensores utilizan SensorML que limita la utilidad de la Nube de FI-WARE, pues está diseñado para definir sensores y actuadores 'sencillos' y no dispositivos más avanzados. Modelar un dispositivo complejo con SensorML, como una cámara, es un proceso complicado[2].
- **Ausencia de capacidades de orquestación:** Debido a que la iniciativa es aún relativamente novedosa y pese a disponer de métodos muy intuitivos (*mash-ups* y *widgets*) para crear aplicaciones web rápidamente, éstas actúan únicamente como sumidero, sin dar opción a realizar tareas de análisis o definir comportamientos.
- **Imposibilidad de acceso a la información de forma local:** La solución basada en FI-WARE no proporciona ninguna forma estandarizada de acceso a los sensores, dejando en manos del implementador la creación de este software, lo cual puede llegar a convertirse en una tarea tediosa debido a la enorme disparidad de dispositivos que se contemplan Internet de las Cosas.

5. Conclusiones y trabajo futuro

En este trabajo se ha presentado una propuesta para tratar de resolver la problemática asociada al desarrollo de una plataforma heterogénea en la que convivan en armonía dispositivos tan diferentes como lo es una pequeña mota de un ordenador tradicional. Los casos de uso planteados en la Sección 3 ilustran nuestra concepción de cómo debe ser un agregador de dispositivos en la Nube en un escenario de Internet de las Cosas.

Actualmente, estamos desarrollando una prueba de concepto denominada DEEP (DPWS *Enabled dEvices Platform* `http://com-gisum-deep.appspot.com/`), en la que se pretenden implementar los mecanismos de orquestación, la cual no se presenta en el artículo por encontrarse en proceso de desarrollo y por cuestiones de espacio.

Como trabajo futuro se plantea la posibilidad de realizar orquestaciones sensibles al contexto, donde la respuesta a una operación dependa de la situación del sistema, dando lugar a múltiples escenarios de orquestación que varían en el tiempo de forma dinámica.

6. Agradecimientos

Este trabajo ha sido desarrollado con el apoyo de los proyectos: TIN2012-35669, financiado por el Ministerio Español de Economía y Competitividad y FEDER;

[1] `http://www.fi-ware.org/`

[2] `http://www.sensorml.com/sensorML-2.0/examples/myHDCamera.html`

FP7-610531-SeaClouds, financiado por la Unión Europea; P11-TIC-7659, financiado por la Junta de Andalucía; y la Universidad de Málaga y el Campus de Excelencia Internacional Andalucía Tech.

Referencias

1. Devices profile for web services version 1.1. OASIS, `http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html` (2009)
2. Cubo, J., Brogi, A., Pimentel, E.: Behaviour-aware compositions of things. In: Green Computing and Communications (GreenCom), 2012 IEEE International Conference on. pp. 1–8. IEEE (2012)
3. Cubo, J., Brogi, A., Pimentel, E.: Towards behaviour-aware compositions of things in the future internet. In: Proceedings of the 2Nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups. pp. 28–35. ACM (2012)
4. Ericsson: More than 50 billion connected devices - taking connected devices to mass market and profitability. White Paper (2011)
5. Forkan, A., Khalil, I., Tari, Z.: Cocamaal: A cloud-oriented context-aware middleware in ambient assisted living. Future Generation Computer Systems 35, 114 – 127 (2014)
6. Karouia, A., Langar, R., Nguyen, T.M.T., Pujolle, G.: Soa-based approach for the design of the future internet. In: CNSR. pp. 361–368. IEEE Computer Society (2010)
7. Laine, M.: RESTful Web Services for the Internet of Things. Ph.D. thesis, Master's thesis, Aalto University School of Science Department of Media Technology (2012)
8. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big'web services: making the right architectural decision. In: Proceedings of the 17th international conference on World Wide Web. pp. 805–814. ACM (2008)
9. Pruett, M.: Yahoo! Pipes. O'Reilly, first edn. (2007)
10. Su, C.J., Chiang, C.Y.: Iaserv: An intelligent home care web services platform in a cloud for aging-in-place. International Journal of Environmental Research and Public Health 10, 6106–6130 (2013)
11. Tapia, D.I., Alonso, R.S., García, Ó., de la Prieta, F., Lancho, B.P.: Cloud-io: Cloud computing platform for the fast deployment of services over wireless sensor networks. In: KMO. vol. 172, pp. 493–504. Springer (2012)
12. Timm, C., Schmutzler, J., Marwedel, P., Wietfeld, C.: Dynamic web service orchestration applied to the device profile for web services in hierarchical networks. In: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE. pp. 18:1–18:6. COMSWARE '09, ACM, New York, NY, USA (2009)

Analysis of the Feasibility to Combine CEP and EDA with Machine Learning using the Example of Network Analysis and Surveillance

Ruediger Gad[1,2], Martin Kappes[2], and Inmaculada Medina-Bulo[1]

[1] University of Cádiz, Spain
inmaculada.medina@uca.es
[2] University of Applied Sciences Frankfurt am Main
{kappes, rgad}@fb2.fh-frankfurt.de

Abstract. Complex Event Processing (CEP) and Event-driven Architectures (EDA) are modern paradigms for processing data in form of events. Machine Learning (ML) methods offer additional sophisticated means for analyzing data. By combining these technologies it is possible to create even more comprehensive and powerful data analysis and processing systems. We analyze the feasibility of combining CEP and EDA with ML using the example of the application domain of computer networks. We present relevant aspects, a sample use case, an sample architecture, and results of performance benchmarks. Our results indicate that the combination of these technologies increases data processing capabilities and that it is feasible from a performance perspective as well.

Keywords: Complex Event Processing, Event-driven Architecture, Machine Learning, Computer Networks

1 Introduction

Complex Event Processing (CEP) and Event-driven Architectures (EDA) offer powerful capabilities for addressing important requirements in distributed, near real-time data processing systems as, e. g., in the field of network analysis and surveillance [1,2,3]. However, often more sophisticated data analysis methods, like Machine Learning (ML), are used as well [4,5].

CEP offers powerful methods for modeling and detecting known relations in real-time data that is processed in form of events [6]. Furthermore, CEP and EDA ease the integration of heterogeneous as well as distributed sensors. ML offers additional sophisticated data analysis capabilities, especially for detecting and analyzing complex, unapparent, and unknown relations in multi-dimensional data [7], which is usually difficult with classical signature based approaches. Moreover, ML techniques offer dynamic and autonomous capabilities such as the capability to learn models from sample data or to use unsupervised methods. With a combination of these approaches the different capabilities can be joined in a complementary way to increase the capabilities of the resulting overall system.

A combination of ML with CEP had already been proposed, e. g., for detecting static CEP rules [8]. We propose a more dynamic integration of CEP and EDA with ML such that the ML component is an equal first class member of the system. While we use the example of computer networks, our results can also be applied to other application domains.

With a sample use case, we show that the combination of CEP and EDA with ML increases the capabilities of the overall system. Benchmark results of the individual components show a data processing rate up to the magnitude of Gigabit Ethernet. Thus, the combination of CEP and EDA with ML results in improved capabilities and is possible from a performance perspective as well.

In the following, we first introduce related work and motivate important requirements. Then we present a sample use case and an architecture for integrating the discussed methods. Afterwards, we present the results of an empirical performance analysis. Finally, we conclude our discussion and present an outlook on future work.

2 Background and Related Work

CEP and EDA are paradigms that were specifically developed for processing data in the form of events [6,9]. One powerful feature of CEP is the ability to correlate events to each other and by this to infer new complexer events from simpler events. The actual processing is performed by a CEP engine which matches event patterns against streams of events. The event patterns used in classical CEP systems are essentially signature-based methods; an event pattern can be seen as a signature that matches specific characteristics of events.

For the challenging task of extracting credible information from increasingly bigger amounts of increasingly multi-dimensional data, ML techniques are more and more used [7]. Instead of relying on pre-defined signatures, ML methods are capable to learn on their own; e. g., it is possible to use a known data set for training an ML algorithm instance which can then afterwards apply the learned knowledge to unknown data. Examples for the application of ML in the field of computer networks are traffic classification [4], IDS [10], or anomaly detection [11]. With ML, different problems like clustering [4] or feature selection [12] can be solved.

A field of study that is also targeted at processing data in form of streams is "data stream mining" [13]. Roughly said, data stream mining tries to create a similar set of methods that are available in "static" data mining for the application to data streams. Aspects discussed in this field are, e. g., window-based interpretation of streams, "one pass" approaches in which each data instance is only processed once, or "concept drift" which describes changes in the data over time. In [14], a detailed discussion of CEP, data stream mining, and other approaches operating on data in form of streams is presented.

We focus our discussion on CEP, EDA, and classical ML. We specifically assess the CEP capability of correlating events with respect to their order to create new event streams containing new features. Additionally, we use classical

ML methods instead of their data stream mining counterparts in order to analyze the feasibility of this approach. Typically, data stream mining methods are even more optimized with respect to throughput and thus can be expected to perform even better.

We use the example of computer network analysis and surveillance for our discussion [15]. While there are different methods for gathering data about a computer network like connection tracking [16] or Security Incident and Event Management (SIEM) systems [17], we use packet capturing [15] for our example.

3 Requirements

In this section, we introduce and motivate the requirements for our approach. While we limit our discussion to the computer networks, the mentioned aspects are applicable to other application domains as well. Computer networks are distributed systems in which data can be acquired with different methods. Thus, they serve as good example for a complex and distributed data analysis system.

For acquiring a good overview of the situation in such a distributed system, data from different locations and with different data acquisition approaches has to be combined. Hence, we propose an approach that enables components to be deployed at different locations and that is capable to process data from heterogeneous sensors.

In our sample scenario, the system components have to be deployed on a wide range of different nodes, like routers, desktops, or servers. Thus, the solution has to be platform independent.

In order to quickly react, it is crucial to obtain information in a timely manner. Consequently, the system has to work in "near real-time".

Finally, our system has to be capable to combine classical CEP and EDA with ML. By this, we intent to leverage the different capabilities from these technologies in a joint manner such that the total capabilities of the resulting overall system are improved.

4 Combining CEP and EDA with ML

The CEP and EDA paradigms inherently fulfill many of the aforementioned requirements. Thanks to the loose coupling in an EDA, the inclusion of heterogeneous data sources and event processing entities is easily possible. EDA and CEP also enable the effortless usage of distributed data sources and allow to create large-scale systems, e. g., by using a multi-tiered architecture [1], and enable the adaption of systems to different situations, e. g., by growing or shrinking depending on the actual requirements [3]. Furthermore, the event-driven processing paradigm allows for near real-time processing.

For ML, the form and selection of the input data is important; the so called features that are analyzed by ML algorithms need to be extracted and preprocessed [12]; e. g., by filters that determine which data is sent to the ML

algorithm or more complex operations that correlate multiple data instances for extracting complexer features.

We use feature extraction and classification based on the extracted features as sample use cases to illustrate ways for meaningfully combining CEP and ML. In the following, we first outline how CEP can be used for feature extraction and present an sample architecture. In the next section, we present benchmark results that were obtained in a feature extraction scenario.

Filtering can be the selection of data instances or properties which are to be sent to the ML part. In our sample scenario, packet capturing data instances could be selected, e. g., depending on the protocol or the header fields and then only the relevant properties could be extracted. In Listing 1.1, a sample event pattern is shown that selects Internet Message and Control Protocol (ICMP) echo request type events and further only extracts the time stamp, IP source and destination address, and the ICMP echo sequence number.

Listing 1.1. Example for Selecting Events and Event Properties

```
SELECT timestamp, ipSrc, ipDst, icmpEchoSeqNo
FROM packet.capture.events WHERE icmpType = "echo request"
```

The correlation of multiple data instances is another important capability of CEP. In Listing 1.2, we show an event pattern that correlates two events with respect to their order as well as depending on their event properties and calculates a new feature based on a property of the two selected events. The followed-by relation "$->$" is used for correlating events "a" and "b" with their order. Furthermore, the relation of events "a" and "b" is checked based on the source, destination, and sequence number properties. As new feature, the difference of the time stamps of "b" and "a" is calculated. The shown pattern extracts the reply times for associated ICMP echo request/reply packets.

Listing 1.2. Example for Correlating Events and Event Properties

```
SELECT
  b.timestamp - a.timestamp, a.source, a.destination
FROM PATTERN [ EVERY
    a = cep.icmp.echo.request ->
    b = cep.icmp.echo.reply(source = a.destination
          AND destination = a.source AND sequence = a.sequence)]
```

The examples in Listings 1.1 and 1.2 illustrate the filtering and inference capabilities of CEP. New features can be extracted that can only be derived by considering more context than a single event. These examples show that CEP offers a rich feature set that complements the capabilities of ML techniques.

In Figure 1, we show an architecture of a CEP system. The flow of events is indicated with arrows. Via the so called "event producers" data is fed into the system in form of raw events. Event consumers are the outputs of the system. Our architecture uses a central communication infrastructure (CI) for interconnecting all components of the system.

In the depicted architecture, the combination of CEP and ML can be achieved in two ways which are indicated with the dashed arrows. Firstly, the ML part can be connected via the central CI. This approach has the advantage that the capabilities of the CI, e. g., for routing events, can be leveraged. A disadvantage

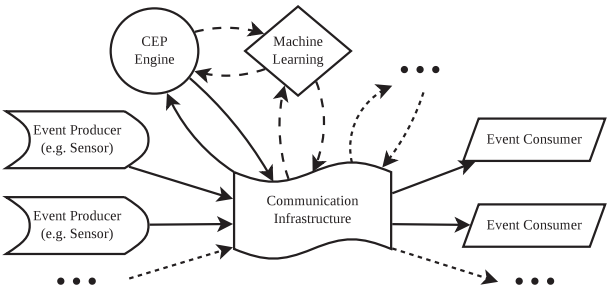

Fig. 1. Architecture for Integrating ML in CEP and EDA

is that a higher load is caused due to the communication overhead. Another way is to directly attach the ML part to the CEP engine. This approach offers a better performance as overhead and latency are reduced. However, features as offered by a CI are missing. Hybrid solutions in which one connection, e. g., from the CEP engine to ML is done directly, whereas the other is realized via the CI are possible as well.

5 Performance Evaluation

Let us now focus on performance aspects of CEP and ML with respect to data throughput. While other aspects like accuracy and detection rates are also important, the data throughput performance is one of the most critical when leveraging commodity hardware for such purposes. In case of insufficient performance, a combination of CEP and ML would practically be impossible.

As one of our goals is platform independence, we used Java-based implementations for our benchmarks. As CEP engine we used the Java Esper CEP engine [18]. As ML implementation we chose the java-ml machine learning library [19].

We use the rate of events that can be processed as performance metric for CEP and ML. For comparing the ML classification algorithms, we distinguish between the learning and the actual classification phase.

With respect to the data rate, packet capturing is typically one of the most resource intensive methods. Thus, it is a straightforward choice in order to determine the limits of our approach. We implemented patterns for deriving three complex features based on packet capturing. They represent typical scenarios in network analysis and surveillance which all require processing more than one event; i. e., they cannot be directly extracted by just applying filters. More precisely, we implemented patterns for extracting the following features: the response time of an ICMP echo request/reply (A), the response time for User Datagram Protocol (UDP) based protocols such as the Domain Name Service (DNS) protocol (B), and the duration of a Transmission Control Protocol (TCP) three-way handshake (C). Additionally, we analyzed whether rate-limiting mechanisms can be applied for increasing the performance.

The hardware we used for the benchmarks was a Laptop with an Intel(R) Core(TM) i5 CPU with a nominal speed of 1.7 GHz (i5-3317U) and 4 GB memory. We explicitly disabled all multi-processor functionality for the benchmarks in order to eliminate artefacts from multi-processor scheduling and the like. As operating system and software environment we used a plain Debian Linux with a standard OpenJDK version 1.7 installation.

For the measurements, we generated a traffic sample containing automatically generated DNS requests, HTTP traffic, and ICMP echo requests/replies. We then used this traffic sample to send events into the CEP engine at a controlled, chosen rate in a loop. For each measurement run, the rate was fixed and the CEP engine was set up to match a single pattern. We measured the CPU usage as well as the rate of events that could be successfully processed. Moreover, we used a warm-up time of 8 seconds before gathering the actually analyzed measurements. The overall run-time of our measurement was 30 seconds; with a warm up of 8 seconds this results in a measured time of 22 seconds.

Figure 2 shows the CPU usage and the successfully processed events for different input event rates for the three patterns when no additional limiting is used. In all scenarios, the rate of successfully processed input events remains near 100% until the CPU usage reaches 100%. Once the CPU is fully utilized, the rate of successfully processed input events decreases as some events are no longer processed due to CPU overload.

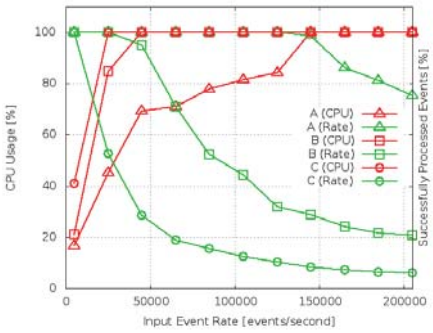

Fig. 2. Performance Results of CEP Feature Extraction

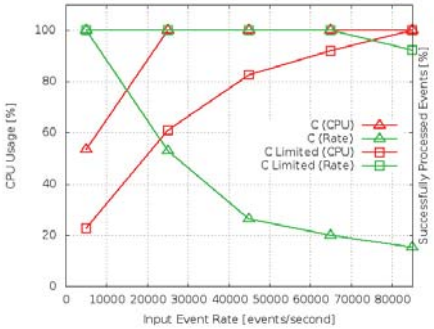

Fig. 3. Performance Comparison for CEP Feature Extraction (TCP)

The event pattern based on ICMP echo request/reply (A) perform best, i. e., a CPU usage of 100% is reached for an input rate of roughly 140,000 events per second (eps), whereas a full CPU utilization is reached earlier for the other patterns. The pattern based on UDP traffic reaches a full CPU utilization for a rate of 40 keps whereas the pattern based on TCP already reaches this point for 5 keps.

Additionally, we tested and compared, for each scenario, a pattern that limits the number of events processed by the pattern via a "every-distinct" clause. Such a clause limits the number of events that are processed by a pattern by removing packets creating duplicate events, i.e., events which have already been generated by other packets.

Figure 3 depicts the results for the TCP scenario. As can be seen, the achievable input event rate is significantly higher when rate limiting is applied. It is possible to successfully process up to 65 keps, whereas, without limiting, the ratio of successfully processed events already sharply drops off from an input rate of 5 keps.

Figure 4 shows the comparison of the results for the UDP scenario. For this scenario, in case of the pattern without limiting the rate of successfully events drops at an input rate of about 45 keps, whereas with limiting 65 keps can be processed. In Figure 5, the comparison of results for the ICMP scenario is shown. In this case both pattern variants perform similarly and reach the maximum at about 140 keps.

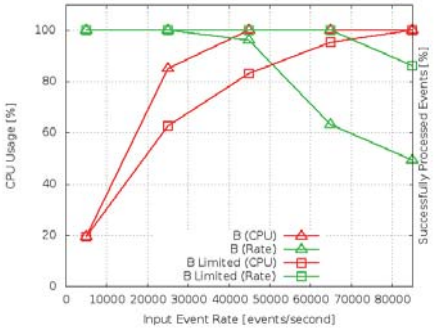

Fig. 4. Performance Comparison for CEP Feature Extraction (UDP)

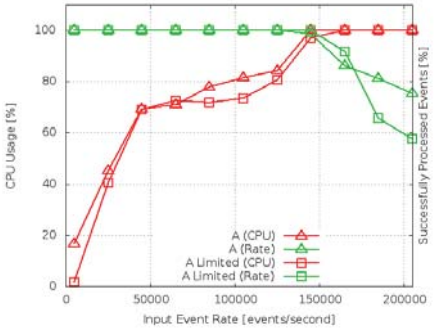

Fig. 5. Performance Comparison for CEP Feature Extraction (ICMP)

For the performance evaluation of the ML algorithms, we classified data with different algorithms. We used a simple scenario aimed on detecting congestion situations in a network based on duration between the occurrence of an ICMP echo request and an ICMP echo reply as main feature. Additionally, we used the TTL values of both packets. While the scenario is not sophisticated, it is suitable for evaluating the ML performance.

For each run, we inserted data over a time period of ten seconds and measured how many data instances had been be processed. We performed an additional ten second run before performing the actual measurement for each learning and classification phase.

Table 1 shows the results for the different algorithms. It shows that the algorithms generally have asymetric characteristics; the performance of the learning and classification phases differ.

Table 1. Performance Comparison of Machine Learning Algorithms

	Performance [data instances/second]	
Algorithm	Learning	Classification
KDtreeKNN	3,045,687	185,671
KNearestNeighbors	11,681,348,465	41,700
LibSVM	1,719,432	4,715,608
NaiveBayesClassifier	846,686	325,264
RandomForest	37,305	166,122
SelfOptimizingLibSvm	40,108	4,589,241

The observed asymetric characteristic of the results can be important for the actual application of the ML algorithms. Depending on the usage scenario the requirement for a ML method may be different. In scenarios where only data is analyzed a fast classification typically is most important. On the other hand, in scenarios that require frequent actualization of learned data a fast learning phase may be more important.

With an Ethernet frame size of 1500 bytes the packet rate of Gigabit Ethernet is about 88,000 packets per second (pps). For smaller sizes, the packet rate further increases, e. g., to roughly 200 kpps for a size of 600 bytes or about 400 kpps for a size of 300 bytes. These rates are, however, the maximum values for the respective frame sizes. In the CEP benchmarks, we observed a throughput of up to 140 keps and in the ML benchmarks the highest observed throughput rates were even higher. Thus, our results show that a throughput up to the magnitude of the data rate for Gigabit Ethernet can be achieved with both CEP and ML. While some results show a throughput smaller than 88 kpps, the majority of results is in this magnitude or even exceeds it. Please also note that we only measured the single core performance and that the used CPU can be considered as low to mid-range class CPU. Consequently, we conclude that combining CEP and ML in the proposed application domain is feasible performance-wise.

6 Conclusion

CEP and EDA offer powerful capabilities for creating large scale, distributed, and flexible data analysis systems. ML techniques enable additional sophisticated operations for the analysis data, e. g., for analyzing big and multi-dimensional data sets that are otherwise hard to handle. A combination of both technologies improves the capabilities of distributed data analysis systems.

At first, we introduced requirements for a distributed, large scale, and flexible data analysis system. Throughout this paper, we used the application domain of

computer networks and network analysis and surveillance as example. Computer networks are distributed systems in which a variety of heterogeneous data can be gathered. Thus, we conclude that our results also provide indicators for the applicability of the combination of CEP and EDA with ML in to other application domains.

We then showed with the help of the application scenario of feature extraction that CEP and EDA can be meaningfully combined with ML with respect to their capabilities. In this scenario, especially the CEP capabilities of correlating multiple events to each other and by this deriving new features are a very good combination in conjunction with ML. With ML, on the other hand, more sophisticated operations can be applied to data. By this, information can be extracted that otherwise would be very hard or impossible to derive with only manually created event patterns and classical CEP. Thus, our results indicate that the combination of EDA and CEP with ML improves the capabilities of data analysis systems.

Furthermore, we presented an architecture for integrating ML in CEP and EDA. The proposed architecture is flexible in order to adapt to different scenarios and implementations.

Finally, we performed multiple performance benchmarks to assess if the individual components perform comparably with respect to throughput as well. Even on commodity hardware and using a performance intensive scenario of packet capturing, the achieved performance was in the magnitude for being applicable in Gigabit Ethernet networks. Hence, our results indicate that the combination of EDA and CEP with ML is practical from the performance perspective as well.

In future work, we plan to further improve the integration of EDA and CEP with ML. In the presented work, we only focused on very general aspects. In subsequent work, we plan to research more detailed aspects and to present more sophisticated solutions as well as perform more detailed empirical analysis. Furthermore, we are going to research ways for combining CEP, EDA, and ML with other stream-based approaches.

Acknowledgement

This work was supported in part by the German Federal Ministry of Education and Research in scope of grant 16BY1201C. Responsible for the content are the authors.

References

1. R. Gad, J. Boubeta-Puig, M. Kappes, and I. Medina-Bulo, “Leveraging EDA and CEP for integrating low-level network analysis methods into modern, distributed IT architectures,” in *VII Jornadas de Ciencia e Ingeniería de Servicios (JCIS - SISTEDES 2012)*, Almería, 2012.
2. G. Lodi, L. Aniello, G. A. Di Luna, and R. Baldoni, “An event-based platform for collaborative threats detection and monitoring,” *Information Systems*, vol. 39, pp. 175–195, Jan. 2014.

3. R. Gad, J. Boubeta-Puig, M. Kappes, and I. Medina-Bulo, "Hierarchical events for efficient distributed network analysis and surveillance," in *Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups*, ser. WAS4FI-Mashups '12. New York, NY, USA: ACM, 2012, p. 5–11.
4. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
5. B. Li, M. H. Gunes, G. Bebis, and J. Springer, "A supervised machine learning approach to classify host roles on line using sFlow," in *Proceedings of the First Edition Workshop on High Performance and Programmable Networking*, ser. HPPN '13. New York, NY, USA: ACM, 2013, p. 53–60. [Online]. Available: http://doi.acm.org/10.1145/2465839.2465847
6. D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.
7. P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge ; New York: Cambridge University Press, Nov. 2012.
8. C. Mutschler and M. Philippsen, "Learning event detection rules with noise hidden markov models," in *2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012, pp. 159–166.
9. O. Etzion, *Event processing in action*. Manning, 2011.
10. L. Wang, S. Zhang, Y. Li, R. Wu, and Y. Yu, "An attribute-weighted clustering intrusion detection method," *Journal of Networks*, vol. 8, no. 6, Jun. 2013. [Online]. Available: http://ojs.academypublisher.com/index.php/jnw/article/view/10214
11. F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency Computation Practice and Experience*, vol. 26, no. 5, pp. 1113–1129, 2014.
12. G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, Jan. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790613003066
13. C. C. Aggarwal, *Data Streams: Models and Algorithms*, 2007th ed. New York: Springer, Dec. 2006.
14. G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Comput. Surv.*, vol. 44, no. 3, p. 15:1–15:62, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2187671.2187677
15. C. Sanders, *Practical Packet Analysis : Using Wireshark to Solve Real-World Network Problems, Second Edition*. No Starch Press, Incorporated, 2011.
16. P. Ayuso, "Netfilter's connection tracking system," *LOGIN: The USENIX magazine*, vol. 31, no. 3, 2006.
17. L. Coppolino, S. D'Antonio, V. Formicola, and L. Romano, "Enhancing SIEM technology to protect critical infrastructures," in *Critical Information Infrastructures Security*, ser. Lecture Notes in Computer Science, B. M. Hämmerli, N. K. Svendsen, and J. Lopez, Eds. Springer Berlin Heidelberg, Jan. 2013, no. 7722, pp. 10–21.
18. Esper contributors & EsperTech Inc., "Esper - complex event processing," http://esper.codehaus.org/, 2014, last accessed 04/14/2014.
19. T. Abeel, "Java machine learning library (java-ml)," http://java-ml.sourceforge.net/, 2012, last accessed 04/14/2014.

SLA4DQ-I8K: Acuerdos a Nivel de Servicio para Calidad de Datos en Intercambios de Datos Maestros regulados por ISO 8000-1x0

Ismael Caballero[1], Isabel Bermejo[1], Luisa Parody[2], Mª Teresa Gómez López[2], Rafael M. Gasca[2] and Mario Piattini[1]

[1]Universidad de Castilla-La Mancha, España
{Ismael.Caballero, Isabel.Bermejo, Mario.Piattini}@uclm.es,

[2]Universidad de Sevilla, España
{lparody, maytegomez, gasca}@us.es,

Resumen. Los datos son uno de los activos más importantes de las organizaciones. Prueba de ello son las iniciativas que están surgiendo para disponer y analizar la mayor cantidad de datos posibles (efecto Big Data), pudiendo así poder descubrir patrones de comportamiento de posibles clientes. Así es frecuente que las organizaciones adquieran datos de terceras partes, datos que son usados como base para los procesos de negocio. Pero, en general, si los datos adquiridos no tienen un nivel de calidad adecuado, entonces no podrá extraerse de ellos el máximo rendimiento. Para evitar esto, es posible establecer acuerdos a niveles de servicio para la adquisición de datos, que es el principal objeto de este artículo. Para ello, se puede usar ISO 8000 partes 100 a 140, que tratan específicamente sobre el intercambio de datos maestros. Para facilitar dicho intercambio de datos, proponemos el uso de un framework que permite combinar los servicios web que satisfacen los correspondientes requisitos de la familia de estándares. Dicho framework consiste en dos componentes: I8K – una arquitectura de servicio – e ICS-API – una interfaz de programación de aplicaciones que permite usar I8K-. La principal aportación de este artículo radica en describir cómo usar el framework para implementar los aspectos tecnológicos de los acuerdos a niveles de servicio, cuando la calidad de datos tiene que ser tenida en cuenta durante el intercambio de datos como parte de la operativa.

Palabras Clave: Calidad de Datos · Acuerdo a nivel de servicio · ISO 8000-1x0 · Intercambio de Datos Maestros · I8K

1 Introducción

Hoy en día la cantidad de datos que mueven las organizaciones es enorme, se calcula que el volumen total de datos que se maneja es de varios petabytes [1]. No obstante, se

está observando un hecho significativo en la forma en la que la demanda de datos se está produciendo. Ya no solo se requiere una gran cantidad de datos, sino que las organizaciones más maduras son conscientes de la necesidad de que los datos que se reciben tengan niveles adecuados de calidad. Por esa razón, cobra sentido la necesidad de llegar a acuerdos a nivel de servicio (SLA), no solo de ciertos aspectos de gestión, sino también correspondiente a ciertos aspectos tecnológicos, en los que se incluyen aspectos estructurales de los datos, y de calidad de los datos [2].

Un acuerdo a nivel de servicio es esencialmente un contrato entre dos organizaciones (una demandante de un servicio, y la otra proveedora del mismo). Dicho contrato contiene un conjunto mínimo de expectativas y obligaciones de ambas partes, recogidos en forma de cláusulas y reglas de actuación que describen cómo proceder cuando las cláusulas no se satisfacen [3]. Las cláusulas que se proponen en este artículo, contendrían requisitos de calidad de datos para los datos que se manejan en el proceso de intercambio de datos. Ejemplos de dichas cláusulas podrían ser: "*C1. Los datos correspondientes a la consulta Q1 deben tener un nivel de compleción superior al 90%",* o "*C2. Los datos tienen que ser devueltos en un tiempo menor que 5 segundos".*

En lo que respecta a las reglas, algunos ejemplos de las decisiones que se pueden tomar son las siguientes: "*R1. Si no se verifica C1, entonces se descartan todos los datos",* o *"R2. Si se verifica C2, entonces se incluyen los datos en el procesamiento".*

En el párrafo anterior, la regla C1 incluía un ejemplo de medición de calidad de datos. Tradicionalmente se ha entendido la calidad de los datos desde el punto de vista de "*fitness for use"* [4], aunque también se ha extendido la visión del "*meeting requirements"* [5]. En la primera visión se dice que los datos tienen calidad, si sirven para el propósito para el que se pretenden usar; en la segunda visión, se entiende que los datos tienen calidad, si satisfacen los requisitos para el que han sido diseñados. Podría ser obvio pensar que si los datos satisfacen dichos requisitos, entonces servirían para el propósito para el que han sido diseñados. El hecho que marca la diferencia entre ambas visiones, es que muchas veces los datos son usados en tareas para las que previamente no habían sido diseñados: por ejemplo, como señala [6], esto ocurre cuando se diseña una base de datos de clientes que han establecido relaciones con la organización, y luego esos datos se usan con fines comerciales. Podría llegar a ocurrir, que si de partida no se decidió añadir el número de trabajadores que tenía la organización cliente, ya que no era relevante para los fines de análisis de compras, cuando se pretende usar con fines comerciales, los datos de los clientes no serán suficientemente válidos para los fines comerciales.

Tanto si se estudia la calidad de los datos desde un punto de vista – *fitness for use* - o desde el otro –*meeting requirements*-, la realidad es que existe una necesidad de medir de forma objetiva y repetible el nivel de calidad de los datos. Esta medición puede hacerse comparando el grado en el que los datos han sido útiles con respecto al uso previsto en el primer caso. En la mayoría de las ocasiones hay que recurrir a la percepción de quien usa los datos, lo que determina subjetivamente la evaluación-. Otra opción es determinando el grado de cumplimiento que tienen los datos implementados con respecto a los requisitos de diseño que se utilizaron en su implementación, lo que en la mayoría de los ocasiones puede hacerse de forma objetiva mediante trazabilidad de requisitos. Además, la evaluación de la calidad de los datos es multidimensional: lo que implica que se necesitan varias dimensiones de calidad de datos. Ejemplos de dichas dimensiones pueden ser la compleción (grado con el que se tienen todos los

valores de los datos que se necesitan), la precisión (grado con el que los datos siguen el formato establecido), la oportunidad (grado con el que los datos están disponibles en el momento en el que tienen que ser usados),… Una clasificación de dichas dimensiones puede encontrarse en [7], o bien en el estándar ISO 25012 [8].
Las partes 100 a 140 de ISO 8000 [9-14] describen ciertos requisitos que se tienen que satisfacer para poder asegurar el nivel de calidad de datos en el intercambio de datos maestros. Los datos maestros representan aquellos conceptos que suponen el conocimiento básico del dominio en el que una organización desarrolla su actividad [15, 16]. Entre los requisitos de ISO 8000-1x0 están: (1) implementar un diccionario de datos con el formato par (identificador, valor) de acuerdo a lo previsto por la parte 110 del estándar; (2) la definición de un conjunto de operaciones para codificar y descodificar los datos de acuerdo a ese diccionario – parte 110-; (3) la necesidad de añadir información sobre el ciclo de vida de los datos, indicando quién, cuándo y qué operación se hizo con los datos, de acuerdo a lo establecido por la parte 120 del estándar -; (4) añadir información sobre la evaluación y certificación del nivel de precisión - parte 130 –; y, (5) añadir información sobre la evaluación y certificación del nivel de compleción – parte 140-.
Derivado de la dificultad que implica la incorporación del estándar dentro de una implementación específica, en [17] se describe un framework con dos componentes: I8K, una arquitectura de servicio que da soporte a estos requisitos; e ICS-API, una interfaz de programación de aplicaciones que permite a los desarrolladores de aplicaciones explotar los beneficios de dicha arquitectura de servicio – hasta la fecha es la única propuesta académica conocida basada en el estándar; en el ámbito industrial está la propuesta de ECCMA (www.eccma.org), aunque esta no incluye todas las características de I8K. Para completar ese trabajo previo, en este artículo se propone cómo se puede usar I8K e ICS-API para facilitar la implementación de los aspectos tecnológicos de los acuerdos a niveles de servicio para los datos, incluyendo su calidad.

1.1. Ejemplo ilustrativo

A fin de ilustrar cómo mediante ICS-API se pueden satisfacer los aspectos tecnológicos, se propone un ejemplo de aplicación. En este caso, una aplicación llamada TripPlanner – que en realidad es un *marketplace* – tiene como objeto del negocio buscar entre sus proveedores aquella combinación de hotel, avión y alquiler de coches más barata satisfaga mejor las necesidades de un determinado usuario. Específicamente, un usuario proporcionará a TripPlanner: una fecha de salida, una de retorno, un punto de partida, un destino y un margen de aceptabilidad. Y TripPlanner, tras sucesivas llamadas a los correspondientes proveedores de datos de vuelos, de hoteles y de alquiler de coches, encontrará la combinación con el coste más barato. Por ejemplo, un usuario pueden establecer "*encontrar un viaje saliendo de Sevilla el día 24 de junio hacia Londres y volviendo el 30 de junio, con una posibilidad irse un día antes y de volverse un día antes o un día después"* y esperará que TripPlanner le devuelva una secuencia de este estilo:

1. Llegará a las 14:45 al aeropuerto de *Gatwik*, y podrá alquilar un coche de la compañía *Drive&Smile* por 20 £ que le llevará al *Hotel Sleep Like an Angel*, donde por 51,34 £ la noche podrá estar hasta el día 30 de junio.

2. El día 30 de junio realice el check out a las 11:30 y alquile un coche con la compañía *RememberToDriveAtLeft* por 21,23£ para ir al aeropuerto de Heathrow donde podrá tomar a las 14:54 el vuelo de vuelta ABC-345 por 98 £ a Sevilla.
3. En torno a las 17.30 podrá alquilar un coche con la compañía *100CaballosPorBanda* por 23, 34 € que le llevará a Santa Justa.

Para poder seleccionar todos estos tramos, TripPlanner tiene que establecer contacto con varios proveedores de vuelos, con varios proveedores de hoteles y con varios proveedores de alquiler de coches. En función de los datos de entrada del usuario, TripPlanner tendrá que montar la combinación de vuelos, alquileres de coche y hoteles que resulten más económica para el cliente.

El resto del artículo está estructurado como sigue: la sección 2 describe brevemente la implementación de referencia I8K e ICS-API. La sección 3 describe algunos elementos que se tienen que incorporar en los SLA4DQ-I8K. En la sección 4 se describe cómo implementar los acuerdos a nivel de servicio usando ICS-API. La sección 5 introduce brevemente conclusiones, y trabajos futuros.

2 I8K e ICS-API: Una implementación de referencia de ISO 8000-1x0

La forma en la que I8K e ICS-API están relacionadas en el framework se muestra en la Figura 1. I8K introduce una colección de funcionalidades basadas en servicios web para dar soporte a ISO 8000-1x0, cuya explotación se puede simplificar haciendo uso de las primitivas de servicios implementadas en ICS-API. Estas primitivas se usarán para implementar y procesar los acuerdos a nivel de servicio de calidad de datos, como explicará en las siguientes subsecciones.

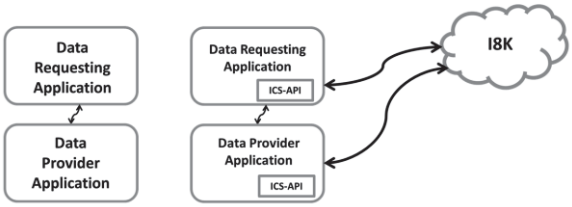

Fig. 1. (a) Forma en la que dos aplicaciones interactúan sin I8K; (b) forma en la que dos aplicaciones interactúan utilizando los servicios de I8K

2.1 I8K: Una arquitectura de servicios para dar soporte al intercambio de datos maestros con certificación de niveles de calidad

Con el objetivo de dar soporte a las partes 100 a 140 de ISO 8000, I8K está compuesta por un conjunto de módulos, cada uno una misión específica, y comunicados mediante servicios Web. Los módulos, con sus correspondientes responsabilidades son: I8KManager (fachada de la arquitectura); IK.Cer110 (responsable diccionario de datos); I8K110 (codificación y decodificación); I8K.Cer130 e I8K.Cer140 (añaden

información sobre evaluación de la precisión y de la compleción respectivamente); y finalmente I8K.Ev130 e I8K.Ev140 (son los que realizan la evaluación de las dimensiones citadas. Para poder gestionar las peticiones se describen una serie de tipos de mensajes de datos maestros (véase Tabla 1).

Tabla 1. Tipos de Mensajes definidos para I8K

Tipo	Descripción
I8K.REQ	Una aplicación manda un mensaje de petición de datos maestros a un proveedor de datos
I8K.RES	Un proveedor de datos devuelve los datos pedidos a la aplicación que la ha solicitado
I8K.COD-GE	La aplicación solicitante necesita codificar los datos maestros para hacer una petición de servicio de datos
I8K.CODED	I8K ha codificado los datos en un mensaje de datos maestros, y el contenido han sido devuelto a la aplicación solicitante
I8K.DEC	La aplicación solicitante o la aplicación proveedora necesita descodificar los datos contenidos en un mensaje de datos maestros que ha recibido para poder entender y procesar el mensaje
I8K.DECODED	I8K ha descodificado el mensaje de datos maestros y el contenido ha sido devuelto a la aplicación que había solicitado descodificar el mensaje
I8K.COD-CR130	Una aplicación necesita codificar los datos maestros contenidos en el mensaje y también certificar el nivel de precisión de los datos contenidos en el mensaje de datos maestros
I8K.COD-CR140	Una aplicación necesita codificar los datos maestros contenidos en el mensaje y también solicita certificar el nivel de compleción de los datos contenidos en el mensaje de datos maestros
I8K.COD-CR	Una aplicación necesita codificar los datos contenidos en el mensaje de datos maestros y certificar el nivel de precisión y compleción.

La operativa habitual de I8K se establece según se describe en la Figura 2.

Por tanto, para comunicar dos aplicaciones asegurando el nivel de calidad de datos, se deben secuenciar los correspondientes tipos de mensajes. Todos los mensajes se encapsulan en una cadena XML, cuyo esquema ha sido diseñado para almacenar no sólo los valores de los datos maestros, sino también la información de los tipos de mensajes y otros aspectos relacionados. En la sección 4 se presentarán algunos detalles relevantes sobre los aspectos de calidad de datos, aunque el lector puede encontrar más detalles sobre la comunicación entre las aplicaciones en [17].

2.2 ICS-API: Una interfaz para facilitar el uso de I8K

A fin de facilitar la programación de los aspectos de intercambio de datos maestros a través de la arquitectura I8K, se desarrolló una colección de primitivas que serán soportadas por la interfaz ICS-API para habilitar las siguientes funcionalidades:

- Especificar información sobre las aplicaciones que participan en el intercambio de datos maestros.
- Especificar la sintaxis y versión de los datos que deben usarse en el intercambio de datos.

- Especificar el nivel de calidad de datos que la aplicación solicitante demanda del proveedor.
- Especificar los términos que van a ser incluidos en el mensaje de datos maestros
- Especificar los requisitos de datos con respecto a los cuales se va a evaluar la calidad de los datos maestros con respecto a la precisión y/o a la compleción

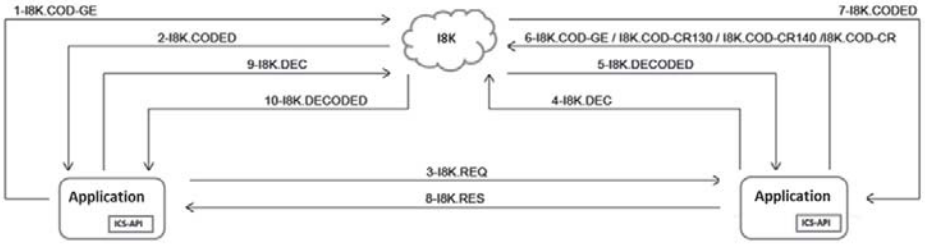

Fig. 2. Operativa de I8K.

3 Elementos en un contrato SLA4DQ-I8K

Los acuerdos a nivel de servicio implican la especificación de ciertos elementos [3, 18]. A continuación se detalla cómo ampliarlos y adecuarlos para un contrato orientado a calidad de datos usando I8K (SLA4DQ-I8K):

- **Aspectos de gestión y organización del acuerdo**
  - Propósito específico de la relación entre solicitante y proveedor: describe el servicio que se presta, que en esta ocasión será del tipo *Data as a Service (DaaS)*, ya que son datos los que se pretende servir.
  - Objetivos detallados del contrato: deberían incluir aspectos de calidad de los datos usados. Como el contrato del servicio que se está definiendo es con respecto a ISO 8000-1x0, entonces las dimensiones de calidad que pueden ser objeto del acuerdo serán la compleción y la precisión.
  - Implicados en el contrato: se deben especificar cuáles serán las responsabilidades de cada uno de ellos. El hecho de tener I8K [17] como elemento identifica tres tipos de organizaciones implicadas: la solicitante de los datos; la proveedora de los datos; y, la responsable de mantener I8K en ejecución y funcionamiento.
  - Coste de la prestación de los servicios: Se incluirá una descripción del coste de los datos servidos, parametrizados por aquellos aspectos que sean de interés a la hora de calcular el coste del servicio: volumen de datos, nivel de calidad de los datos, tiempo de respuesta, calidad del servicio (que a su vez puede influir y verse influida por el nivel de calidad de los datos). En los costes hay que incluir el del servicio proporcionado por I8K.
  - Frecuencia de revisión del contrato: veces que hay que revisar el contrato, y cómo y cuándo resolverlo.

- **Aspectos tecnológicos del acuerdo sobre los datos que son objeto del intercambio de datos:**
  - Descripción de los datos que deben ser intercambiados: p.e. aquellos que satisfacen una consulta predefinida específica. En el caso que nos ocupa, como I8K está centrado en datos maestros, entonces será necesario identificar las entidades de datos maestros que forman parte del acuerdo.
  - Formato: representado por la sintaxis y la versión de los datos que tienen que ser intercambiados.
  - Canal de comunicación: representando cómo los datos serán solicitados y cómo los datos serán servidos, p.e. a través de servicios Web, o de *endpoints* que proporcionan los datos. Además, en caso de ser necesario, se deberá establecer la forma de acceder programáticamente a los servicios.
  - Esquema de acceso a los datos: los datos solo pueden ser consultados por un usuario específico en un intervalo de tiempo determinado, o sólo pueden realizar un número finito de puentes en la comunicación.
  - Nivel de calidad aceptable de los datos, con respecto a la precisión y la compleción, p.e. 90% para precisión y 50% para compleción.
  - Costes de los datos y penalizaciones en caso de no alcanzar los niveles de calidad: p.e. cada llamada a los servicios de los datos cuesta 0,10 €; en caso de que no se satisfagan los niveles mínimos aceptables de calidad, no se pagará la llamada.

4 Uso de ICS-API para la implementación de los aspectos tecnológicos del SLA en un ejemplo de aplicación

A continuación se detalla cómo la propuesta presentada se aplica al ejemplo ilustrativo introducido en la sección 1.1. El interés se centra en la relación que TripPlanner tiene que establecer con cada proveedor de datos, con los que tiene que firmar un Acuerdo a Nivel de Servicio. Este acuerdo a nivel de servicio, como se comentó en el apartado anterior, tiene unos aspectos de gestión y otros aspectos tecnológicos. Si los diseñadores de las correspondientes aplicaciones – tanto de TripPlanner como de cualquiera de los otros proveedores de datos- quieren implementar los acuerdos a nivel de servicio incluyendo los aspectos de calidad de datos, pueden optar por utilizar I8K e ICS-API. La aportación de este artículo es: "demostrar cómo usando ICS-API se pueden implementar los aspectos tecnológicos de los acuerdos a nivel de servicio incluyendo aspecto de calidad de datos". Como resultado de dicha implementación, se creará por cada uno de los escenarios el correspondiente mensaje de datos maestros, que será descrito usando XML – como se indicaba en la sección 2.1-. Los escenarios condicionan el tipo de acciones que se puede realizar, y por tanto la elección de los tipos de mensajes de datos maestros representados en la Tabla 1.

Siguiendo el ejemplo que se ha propuesto, se anima al lector a centrarse en el intercambio que TripPlanner hará con uno de los proveedores de datos de vuelos, llamado FlightCIA. La secuencia de acciones que un desarrollador de TripPlanner tiene

que seguir para pedir datos a FlightCIA vendrá determinada por el acuerdo a nivel de servicio que haya sido establecido, y que se representa en la Tabla 2.

Cuando el desarrollador de TripPlanner esté particularizando mediante ICS-API el código para hacer una llamada de tipo de mensaje I8K.REQ escribirá el siguiente trozo de código para generar la correspondiente cabecera del mensaje de datos maestro:

```
MDQManager manAP=new MDQManager();
manAP.configureOrganization(''TripPlanner'',
OrganizationType.AP);
manAP.setDestination(''FlightCIA'');
…
manAP.configureCertification(0, 50, false, true);
```

que generará la siguiente cabecera del Mensaje de datos maestros:

```
<head>
        <type-message type="I8K.REQ"/>
        <syntax syntax_version="1.0" syntax_name="Classical" syntax_id="1"/>
        <cert130 requiredlevelthreshold130="0" certificated130="false"/>
        <cert140 requiredlevelthreshold140="50" certificated140="true"/>
</head>
```

Tabla 2. SLA4DQ-I8K entre TripPlanner y FlightCIA

Elemento	Valor
Sintaxis a usar para el intercambio	Classical
Versión Sintaxis	1.0
Certificación ISO 8000-130 necesaria	No
Certificación ISO 8000-140 necesaria	Sí
Umbral mínimo de aceptación para ISO 8000-130	0
Umbral mínimo de aceptación para ISO 8000-140	50
Listado de Términos Requeridos	"FLIGHT_PRICE", "CARRIER"
Patrones de Precisión para CHECK_IN	((19|20)dd)-(0?[1- 9]|1[012])-(0?[1-9]|[12][0-9]|3[01])$"
Fuentes confiable de Datos para DESTINATION	http://wordnet.princeton.edu/
Acción si no se supera nivel de calidad	Solicitar nuevos datos

A medida que se van completando la creación del mensaje de datos maestros se irá obteniendo el XML. Así

```
manAP.addTermAndValue("FROM_LOCATION", "Seville");
manAP.addTermAndValue("TO_LOCATION", "London");
…
message=manAP.encodeAndCertificated();
```

permitirá obtener, tras la codificación de los términos indicados, la siguiente porción de código XML en el cuerpo del mensaje:

```
<data>
        <property-value property-ref="01-02#00-000008#1">
                <controlled-value value-ref="2014-06-22"/>
                <controlled-value value-ref="2014-06-23"/>
        </property-value>
                        …
        <property-value property-ref="01-02#00-000001#1">
                <controlled-value value-ref= "Seville"/>
        </property-value>
</data>
```

En el código XML anterior es preciso resaltar el efecto de la codificación: el término "FROM_LOCATION" se sustituye por el identificador que se le ha asignado en el diccionario de datos "01-02#00-000008#1".

Y en lo que respecta a los aspectos específicos de calidad de datos, usando las siguientes primitivas de ICS-API (con los valores de la Tabla 2):

```
manAP.addTermRequired(''FLIGH_PRICE'', true);
manAP.addTermRequired(''CARRIER'', true);
manAP.addTermPattern(''CHECK_IN'', "^((19|20)dd)-(0?[1-
9]|1[012])-(0?[1-9]|[12][0-9]|3[01])$", true);
manAP.addTermSource("DESTINATION",
"http://wordnet.princeton.edu/", true);
```

generará la siguiente información en el mensaje de datos maestros:

```
<data-quality-rules>
    <cert130>
        <set term="01-02#00-000013#1" pattern="((19|20)dd)-(0?[1-9]|1[012])-(0?[1-
        9]|[12][0-9]|3[01])$" required="true"/>
            <set term="01-02#00-000025#1" source="http://wordnet.princeton.edu/"
            required=true
    </cert130>
        <cert140>
            <set term="01-02#00-000024#1" dqproperty="required"/>
            <set term="01-02#00-000022#1" dqproperty="required"/>
        </cert140>
</data-quality-rules>
```

Este mensaje será enviado a FlightCIA, que generará los datos correspondientes a la consulta realizada, siendo similar para el resto de datos y servicios. Con el resultado, se generará un nuevo mensaje de datos maestros del tipo I8K.RES que habrá completado el ciclo correspondiente de uso con I8K. Al final, TripPlanner tendrá los datos en su poder, en el que se ha establecido el correspondiente acuerdo. Para poder usarlos, TripPlanner, mediante las primitivas de ICS-API relacionadas con la aprobación de los niveles de calidad, procederá a incorporarlos al procesamiento de los datos (podría ser, por ejemplo, el que generará el valor para el vuelo ABC-345) o bien desecharlos si no tienen el nivel adecuado de calidad.

```
If(manAP.getLevel130provided()and
manAP.getLevel140provided())
     ShowDataToUser();
else
     throw new DataQualityException(''PedirDatosOtraVez'');
```

5 Conclusiones y trabajos futuros

La principal novedad de este artículo es la de mostrar cómo usar ICS-API para implementar los aspectos tecnológicos relacionados con la calidad de los datos en los mensajes de intercambios de datos maestros. El uso de ICS-API viene derivado de la necesidad de incorporar aspectos de calidad de datos, como los propuestos en el estándar ISO 8000, dentro de los contratos de acuerdos a nivel de servicios.

Como trabajo futuro nos planteamos tratar con otras dimensiones de calidad de datos, que aunque no estén observadas en el estándar, son susceptible de formar parte de un SLA4DQ-I8K.

Acknowledgments. Este trabajo forma parte de los trabajos realizados en el proyecto de investigación GEODAS (TIN2012-37493-C03-01), financiado por el Ministerio de Economía y Competitividad, y el Fondo de Desarrollo Regional, FEDER; asimismo está soportado por TDiaCO-BPMS (TIN2009-13714), financiado por el Fondo de Desarrollo Regional (ERDF/FEDER), y por el V Plan de Investigación de la Universidad de Sevilla (VPPI-US).

References

1. Bertolucci, J.: Think Today's Data Is Big? Wait 10 Years. InformationWeek, http://www.informationweek.com/big-data/big-data-analytics/think-todays-data-is-big-wait-10-years/d/d-id/1113883 (2014)
2. Loshin, D.: The practitioner's guide to data quality improvement. Elsevier (2010)
3. Ward, C., Buco, M.J., Chang, R.N., Luan, L.Z.: A generic SLA semantic model for the execution management of e-business outsourcing contracts. E-Commerce and Web Technologies, pp. 363-376. Springer (2002)
4. Wang, R.Y.: A Product Perspective on Total Data Quality Management. Communications of the ACM 41, 58-65 (1998)
5. Benson, P., Hildebrand, M.: Managing Blind: A Data Quality and Data Governance Vade Mecum. ECCMA, Bethlehem (Pensylvania) (2012)
6. Loshin, D.: Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph. Elsevier (2013)
7. Strong, D.M., Lee, Y.W., Wang, R.Y.: Data quality in context. Communications of the ACM 40, 103-110 (1997)
8. ISO: ISO/IEC 25012:2008 - Software engineering. Software product quality requirements and evaluation (SQuaRE). Data quality model International Organization for Standarization (2009)
9. ISO: ISO/DIS 8000-100: Master Data: Exchange of characteristic data: Overview. (2009)
10. ISO: ISO/IEC 8000-102: Master Data: Exchange of characteristic data: Vocabulary. (2009)
11. ISO: ISO/IEC 8000-110: Master Data: Exchange of characteristic data: Syntax, semantic encoding, and conformance to data specification. (2009)
12. ISO: ISO/DIS 8000-120: Master Data: Exchange of characteristic data: Provenance. (2009)
13. ISO: ISO/DIS 8000-130: Master Data: Exchange of characteristic data: Accuracy. (2009)
14. ISO: ISO/DIS 8000-140: Master Data: Exchange of characteristic data: Completeness. (2009)
15. Borek, A., Parlikad, A.K., Webb, J., Woodall, P.: Total Information Risk Management: Maximizing the Value of Data and Information Assets. Newnes (2013)
16. McGilvray, D.: Executing Data Quality Project: Ten Steps to Quality Data and Trusted Information. Morgan Kaufmann, Burlington, MA, USA (2008)
17. Caballero, I., Bermejo, I., Parody, L., Gómez-López, M.T., Gasca, R.M., Piattini, M.: I8K: An Implementation of ISO 8000-1x0. 17th International Conference on Information Quality (ICIQ), Little Rock, AR, USA (2013)
18. Wustenhoff, E.: Service Level Agreement in the Data Center. Sun Blueprints. Sun Microsystem, http://www.sun.com/blueprint (2002)

Una revisión de la notación PPINOT para indicadores de rendimiento mediante su aplicación a un caso real

M. Cruz, B. Bernárdez, A. del-Río-Ortega, A. Durán

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Sevilla
{cruz, beat, adeladelrio, amador}@us.es

Resumen. Cada vez son más numerosas las organizaciones orientadas a procesos que, para conseguir sus objetivos, necesitan modelar los procesos de negocio (PN) y evaluar su rendimiento. El modelado de procesos hace posible el control del proceso durante su ejecución permitiendo su posterior análisis y mejora. Un elemento clave para llevar a cabo esa mejora son los indicadores de rendimiento de proceso PPIs (*Process Performance Indicators*) que proporcionan información sobre la ejecución del proceso (por ejemplo medidas de tiempo o del estado de los elementos del proceso). La notación PPINOT permite representar de forma gráfica dichos indicadores sobre los elementos del PN y calcular su valor a partir de los datos generados durante la ejecución del proceso. Además de la notación gráfica, existe una notación basada en plantillas que permite la definición textual de los PPIs. En este artículo se presenta una revisión de PPINOT mediante su aplicación a un proceso híbrido (en parte humano, en parte automatizado). Para ello se ha modelado con BPMN el proceso de gestión de correo electrónico del SIC (Servicio de Informática y Comunicaciones) de la US (Universidad de Sevilla) y se han especificado en PPINOT parte del panel de indicadores definido por esta organización. Tras la especificación, se ha visto que la notación se adapta a este tipo de procesos y se han identificado algunas posibles mejoras en la misma.

Palabras clave: indicadores de rendimiento de proceso; indicadores clave de rendimiento; gestión de procesos de negocio; rendimiento de procesos.

1 Introducción

Hoy en día, muchas empresas están adoptando una perspectiva orientada a procesos en su negocio como *una forma de identificar los pasos que realmente crean valor, quién está involucrado en el proceso y cuál es la información intercambiada, es decir, encontrar la manera de mejorar, aumentar la calidad, reducir pérdidas y ahorrar tiempo* [3]. Según [10], la gestión de los PNs (*Business Process Management*) incluye métodos, técnicas, y herramientas para diseñar, aprobar, controlar y analizar los procesos operativos.

El modelado de PN se realiza, generalmente, sobre procesos desarrollados por personas o sobre procesos híbridos en los que se mezclan actuaciones humanas y auto-

matizadas. En nuestro caso, se trata de un proceso prácticamente automatizado que se ofrece a los clientes del correo como servicio, lo que se conoce como PRaaS (*Process as a Service*).

Además, la medida del rendimiento de los procesos es un aspecto esencial en cualquier organización orientada a procesos para la consecución de sus objetivos operacionales y tácticos [1]. Según [2], cada proceso se debería medir con unas métricas para caracterizar las bases de su rendimiento. A dicha medida se le denomina indicador clave de rendimiento KPI (*Key Performance Indicator*). Los KPIs relacionados con el proceso y que se pueden definir con métricas cuantificables se conocen también como indicadores de rendimiento de proceso PPI (Process *Performance Indicator*) y permiten evaluar la eficiencia de los procesos [8].

La notación PPINOT, propuesta en [6], permite la representación gráfica de los PPIs que se hayan definido sobre el propio PN. Además, se ha desarrollado una notación para la definición textual de los PPIs basada en plantillas lingüísticos [8] que ayuda a estructurar la información asociada a cada uno de los indicadores.

El objetivo principal de este trabajo es presentar una revisión de la notación y las plantillas PPINOT al aplicarlas a un caso real. Se tomará como escenario de aplicación lo relativo al proceso de recibir un correo electrónico de una cuenta externa a la US de algún miembro de la comunidad universitaria. La notación utilizada ha sido BPMN 2.0 usando ORIX – herramienta para el modelado de procesos en la web [13] – a la que se ha incorporado un plugin denominado PPINOT-*Modeler* [14][15] que permite representar los PPIs en el diagrama de PN utilizando la notación PPINOT.

El resto del artículo se organiza de la siguiente manera. En la sección 2 se hace una breve introducción a la notación PPINOT, plantillas y patrones lingüísticos, en la sección 3 se presenta el escenario de aplicación sobre el que se va a definir los indicadores y revisar la notación y en la última sección se resumen las conclusiones obtenidas y trabajo futuro.

2 Background

Los PPIs permiten evaluar la eficiencia y eficacia de los PNs; pueden ser medidos directamente a través de los datos que se generan dentro del flujo del proceso y tienen como objetivo la optimización continua y controlada de los procesos [7].

Respecto a la relación entre KPIs y PPIs, hay autores que utilizan indistintamente los términos, otros consideran los PPIs en el nivel operacional y los KPI en los niveles táctico y estratégico [7]. En este trabajo de acuerdo con [1][4], vamos a considerar que los PPI son un caso particular de los KPIs definidos para medir el rendimiento de los PNs. Los PPIs se calculan durante la ejecución del proceso mientras que los KPIs son una forma de medir los objetivos de la organización.

Como todo KPI, el conjunto de criterios que hacen adecuado un PPI para su posterior análisis se conoce como condiciones SMART [12] que se corresponde con la abreviatura de cinco palabras: Específico (*Specific*); Medible (*Measurable*); Alcanzable (*Achievable*); Relevante (*Relevant*) y Tiempo limitado (*Time-bounded*).

2.1 Notación PPINOT

PPINOT [1][5][6] trabaja sobre una o varias instancias de PN que se ejecutan de forma simultánea a lo largo del tiempo. En esta situación PPINOT mide el rendimiento haciendo cálculos que se basan en el flujo de proceso de dichas instancias, es decir, en el camino del PN que siguen las instancias de los mismos.

Si la métrica se define para una sola instancia, o ejecución del proceso, usaremos una *Base Measure* (medida base). Si hay varias, se usará una *Aggregated Measure* (medida agregada) que permite agregar las instancias mediante una función de agregación que por defecto es SUM aunque admite MIN, MAX o AVG.

Existen además las *Derived Measures* (medidas derivadas) que realizan una función matemática sobre varias medidas.

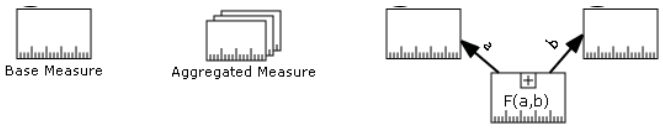

Fig. 1. *Base Measure, Aggregated Measure y Derived Measure* en PPINOT

Las medidas que se pueden tomar son:

- *Time Measure* (tiempo): cálculo de la duración entre dos condiciones de instantes.
- *Count Measure* (conteo): número de veces que cierta condición de instancia se cumple.
- *State Condition Measure* (medida condición de estado): comprueba si una actividad, *pool* o ciertos eventos están en un estado dado (cancelado, activa, etc.).
- *Data Property Condition Measure* (medida condición propiedad de dato): comprueba si se cumple una condición sobre una propiedad de un dato.
- *Data Measure* (medida de dato): permite obtener el valor de la propiedad de un dato de un objeto de datos.

Fig. 2. Qué medir según notación PPINOT

Todas las medidas se pueden utilizar como medidas básicas o como agregadas.

Los distintos tipos de medidas (*Base*, *Aggregated* y *Derived Measure*) utilizan diferentes conectores dependiendo de lo que se vaya a medir. Es decir, el PPI se enlaza a alguno de los elementos del modelo propuesto (actividad, evento, *pool* u objeto de dato) sobre el que se hace la medición y tiene la información necesaria [1].

2.2 Plantillas PPINOT

Como alternativa a la definición gráfica de los PPIs, PPINOT incluye las plantillas [8] que estructuran la información asociada a cada indicador en forma de tabla. Mientras que la notación gráfica parece transmitir menos información (aunque se puede completar usando los atributos definidos en la propia notación que no aparecen visualmente), la notación textual permite dar toda la información: qué instancias participan en el cálculo, cuándo se hace la medida y durante cuánto tiempo se mide o su valor objetivo (límite). Además, las plantillas no requieren un aprendizaje previo a diferencia de la notación gráfica.

Dentro de las plantillas se usan sus patrones lingüísticos [8] que permiten normalizar la redacción de algunos campos de la plantilla facilitando su descripción. Rellenar espacios en blanco en oraciones prescritas es más rápido y menos propenso a errores al seguir un esquema común. Este enfoque ha sido ya utilizado en áreas de Ingeniería de Requisitos [9].

La tabla 1 muestra la plantilla para un PPI, tiene nombre, descripción y una serie de atributos con sus respectivos valores. En cada descripción hay partes fijas (patrón) y partes a cumplimentar, marcadas con < >. Por simplicidad, el atributo alcance – que permite filtrar las instancias del proceso y especificar cuándo se calcula el PPI – se describe en la plantilla aunque en la propuesta original se define aparte.

Table 1. Plantilla para la especificación textual de PPIs

PPI–<id>	<Descripción del nombre del PPI>
Proceso	<Nombre del proceso (Id del proceso)>
Objetivos de negocio	<Objetivos estratégicos y operacionales relacionados con el PPI>
Medida del indicador	El indicador se mide como <Medida>…
Valor objetivo	El valor del indicador debe ser <Valor objetivo>
Alcance	Las instancias del proceso consideradas para el PPI son o Todas o [Aquellas en las que] <Condiciones y periodicidad>
Origen	<Fuente desde la que se obtiene la medida>
Responsable	{<role >|<departamento >|<organización >|<persona>}
Destinatario	{<role >|<departamento >|<organización >|<persona>}
Comentarios	<Comentarios adicionales acerca del PPI>

3 Escenario de aplicación

El SIC de la US proporciona a los miembros de la Comunidad Universitaria (Personal docente e investigador, personal de administración y servicios y alumnos) un servicio de correo electrónico institucional que les permite disponer de una cuenta de correo electrónico accesible vía web, o mediante un cliente de correo ya sea con el protocolo POPS o IMAPS.

El SIC está desarrollando un Panel de Indicadores de Servicios en el que se definen los indicadores de rendimiento que desean conocer y sirven de base para este estudio.

Tras modelar en BPMN el proceso de recepción de correo del SIC, se han especificado los PPIs del Panel de Indicadores para estudiar a) si la notación PPINOT permite representar todos sus indicadores y b) la facilidad de uso y comprensión de dicha notación por parte de usuarios finales y técnicos.

Para poder llevar a cabo el trabajo, se fijó un protocolo de actuación y se han mantenido varias reuniones con personal experto del SIC del área de Apoyo a la Docencia e Investigación.

3.1 Proceso de negocio de recepción del correo en el SIC

El servicio de correo de la US está integrado por múltiples sistemas: estafetas primarias de recepción de mensajes, sistemas antivirus/*antispam*, repartidores de correo electrónico y los buzones en sí mismo.

La estafeta de entrada es responsable tanto de la atención al MX de correo (*Mail Exchange*), es decir, un correo que venga desde un MTA (*Mail Transfer Agent*) de Internet (por ejemplo Gmail, Hotmail, Yahoo) como de recoger correo procedente de un cliente de correo (por ejemplo Thunderbird, Outlook) o dispositivos móviles.

En caso de que el correo sea MX – no autenticado – se comprueba la IP fuente para ver si está en una lista negra de reputación RBL (*Real-time Blackhole Lis*t).

Adicionalmente, se hacen las siguientes comprobaciones: que el dominio del correo destino sea @*us.es – solo se comprueba para correo no autenticado –; el tamaño del correo <= 20 Mb; tiene menos de 100 destinatarios; la dirección IP origen/usuario no ha superado los umbrales máximos de envío de correo; el receptor existe. Si falla alguno de estos chequeos, se emite un código de error SMTP (*Simple Mail Transfer Protocol*) indicando al MTA remoto que no se acepta el correo y es su responsabilidad notificar adecuadamente al emisor; y se aplican los sistemas de filtrado antivirus/*antispam.*

Superados los controles, el mensaje entra en el buzón del receptor, comprobando si la dirección es un alias y que no se supera la cuota de disco con este mensaje.

3.2 Indicadores de rendimiento KPI/ PPI

Una vez analizados los indicadores propuestos por el SIC, se han clasificado según su naturaleza en PPIs, los que se miden cuando hay instancias del proceso en ejecución y KPIs los demás.

Los KPIs identificados son: número de buzones, número de buzones por colectivo (alumos, pdi, etc.), número de reglas *antispam*, número de reglas globales, número de reglas personalizadas, número de reglas de lista blanca personalizada, número de reglas de lista negra personalizada y número de estafetas secundarias.

En la tabla 2 se muestran: el identificador asignado al PPI, su descripción, el tipo de medida para su cálculo según lo descrito en la sección 2 y qué se mide.

Table 2. Relación de PPIs del proceso recibir un correo electrónico del SIC

Cód. PPI	Descripción	Tipo Medida	Qué medir
PPI-Gb	Cantidad total de Gb transferidos	*Aggregated*	*Data*
PPI-Virus	Nº de mensajes con virus	*Aggregated*	*Count*
PPI-Email	Nº de mensajes recibidos	*Aggregated*	*Count*
PPI-POP	Nº de accesos por protocolo POP	*Aggregated*	*Count*
PPI-IMAP	Nº de accesos protocolo IMAP	*Aggregated*	*Count*
PPI-Web	Nº de accesos con navegador	*Aggregated*	*Count*
PPI-RBL	Nº de mensajes descartados por RBL	*Derived Measure*	*Count*
PPI-Spam	Nº de mensajes marcados como *spam*	*Aggregated*	*State Condition*

3.3 Modelo de proceso y PPIs definidos

Para representar el proceso en BPMN se han desarrollado dos modelos. El primero de ellos recibir correo (RC-SIC) representa qué hace el servidor de correo de la US cuando llega un *email* (fig. 3). El segundo, leer correo (LC-USR) representa qué hace el usuario durante una sesión de lectura de correo (fig. 4).

Se han considerado dos procesos distintos porque el proceso RC-SIC afecta a un solo *email* y en el proceso LC-USR se pueden procesar varios *emails* a la vez. Además, el tratamiento de cada *email* en el servidor de correos de la US está totalmente automatizado, mientras que el proceso LC-USR es prácticamente humano.

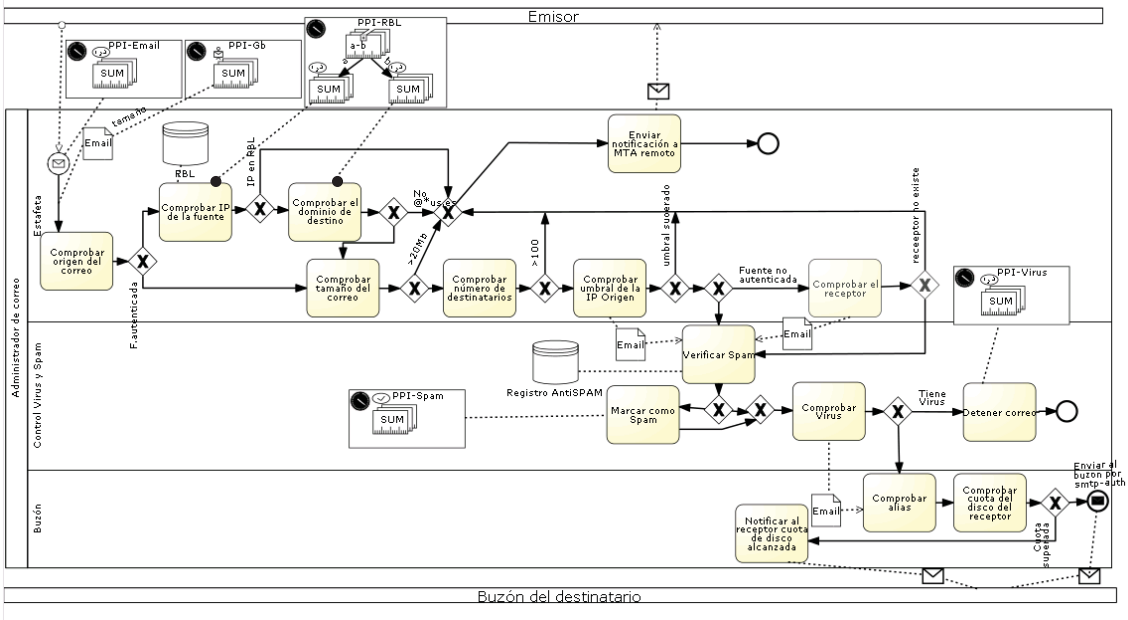

Fig. 3. Modelo de proceso RC-SIC junto con los PPIs definidos

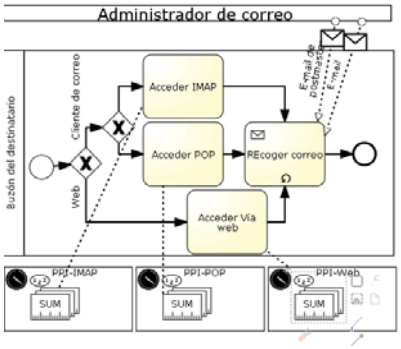

Fig. 4. Modelo de proceso LC-USR junto con los PPIs definidos

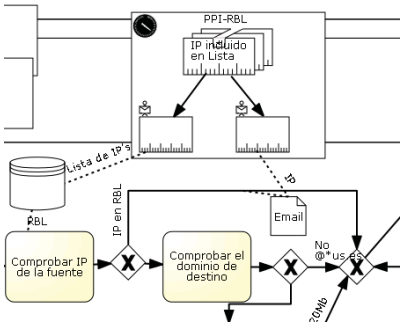

Fig. 5. Definición del PPI-RBL con *Derived Measure* sobre objeto de datos

En las siguientes secciones se discute cada uno de los PPIs especificados. Se han clasificado según el tipo de medida que se efectúa para su cálculo.

3.3.1 Revisión del indicador de tipo *Data Measure*

El indicador PPI-Gb se ha definido como *Data Measure* sobre el objeto de datos *email*. La medida se efectúa al agregar las instancias de dicho objeto y sumar el campo tamaño del *email* en él.

Para definir este indicador, se ha representado explícitamente el *email* que según BPMN se podría asumir que va implícito a lo largo de todo el flujo de proceso. A continuación se muestra la especificación del PPI-Gb usando la plantilla.

Table 3. Plantilla para la especificación del PPI-Gb

PPI–Gb	Cantidad total de Gb transferidos
Proceso	RC-SIC Recibir correo
Objetivos de negocio	Prever las necesidades de dimensionamiento futuro del sistema
Medida del indicador	El indicador se mide como suma del tamaño de todos los *emails*
Valor objetivo	No definido
Alcance	o Todas las instancias del proceso en estado *completed* o Periodicidad mensual
Origen	Logs de eventos
Responsable	SIC
Destinatario	SIC

3.3.2 Revisión de los indicadores de tipo *Count Measure*

Los indicadores PPI-Virus, PPI-POP, PPI-IMAP, PPI-Web han sido definidos mediante *Count Measure* sobre una actividad.

El indicador PPI-Email también se ha definido mediante *Count Measure* pero en este caso sobre el evento de inicio de la recepción ya que se desea contar el número de

veces que se produce el evento de recibir un *email*. Definir un indicador sobre un evento está contemplado en la notación. El resto de los indicadores calculados mediante *Count Measure* se han modelado sin complejidad.

3.3.3 Revisión del indicador de tipo *State Condition Measure*

El indicador PPI-Spam se ha definido usando una *State Condition Measure* sobre la actividad 'Marcar como *Spam*' cuando el estado de dicha actividad sea *completed*. Más tarde, se observa que es posible representarlo mediante *Count Measure*. Es decir, es equivalente sumar las veces que se realiza una tarea o actividad (*Count Aggregated Measure* sobre una actividad) y sumar las veces que una actividad está en estado *completed* (*State Condition Aggregated Measure* sobre una actividad).

3.3.4 Revisión del indicador de tipo *Derived Measure*

Para calcular el valor del PPI-RBL se han barajado dos opciones. La primera de ellas (fig. 3) es definir el indicador mediante *Derived Measure* que calcula la diferencia entre dos PPIs (a-b), definidos sobre dos actividades, por la primera actividad fluyen todos los *email* a validar y por la segunda ya no fluyen los *email* descartados por RBL. La otra opción correcta (fig. 5), es también mediante *Derived Measure* pero en este caso se comprueba si la IP del *email* está incluida en las IPs que integran la lista negra de reputación RBL. Ha sido necesario hacer explícito el objeto de datos *email* en el flujo de secuencia poder definir el indicador sobre él.

En definitiva, existen dos posibles maneras de representar un mismo PPI en el modelo, lo cual ocurre también en otras ocasiones y está contemplado en PPINOT.

3.4 Discusión

El Panel de Indicadores del SIC para el proceso de correo consta de 16 indicadores de rendimiento, de ellos 8 son KPIs y 8 PPIs. De los PPIs identificados, siguiendo PPINOT, todos se definen mediante *Aggregated Measure* o *Derived Measure* y ninguno mediante *Base Measure* que actúa sobre una única instancia. Habría que estudiar si el resultado es extrapolable a la definición de indicadores en otras organizaciones.

En este proceso no se han utilizado *Time Measures*, suelen usarse en procesos humanos que requieran el cálculo de plazos, aunque también tendrían sentido en proceso automatizados, para medir tiempos de respuestas de las máquinas, etc.

Todos los PPIs del SIC hemos podido representarlos con la notación y 6 de ellos de dos formas distintas. Esto ya estaba previsto en PPINOT aunque inicialmente puede generar cierta incertidumbre. En el caso de *Count Measure* y *State Condition Measure*, su uso es equivalente en caso de medidas agregadas definidas sobre una actividad cuando la condición para la actividad de *State Condition Measure* sea estado *completed*. En un caso cuenta el número de veces que se ejecuta una actividad y en el otro suma las actividades finalizadas. No son equivalentes cuando las medidas no están agregadas, cuando el estado de la actividad es distinto de *completed* o cuando se trata de medir sobre un objeto de datos – en ese caso se utiliza *Data Measure* –. En consecuencia los indicadores PPI-Virus, PPI-POP, PPI-IMAP, PPI-Web, PPI-RBL

que inicialmente se especificaron mediante *Count Measure*, también se podían haber especificado como *State Condition Measure*.

Además, se ha visto que en determinadas situaciones hay que hacer explícito algún elemento del modelo de procesos – por ejemplo, un objeto de datos – para poder representar un PPI concreto, porque BPMN asume implícitamente una semántica en el flujo de proceso que se evidencia al especificar los PPIs.

Por otro lado, la definición de los PPI con plantillas ha resultado útil porque además de aportar información alternativa a la notación gráfica, evita la sobrecarga del modelo gráfico – esto puede solucionarse mediante software, ocultando los PPIs que no se quieran mostrar, usando ajustes para escalar, etc. – y más importante, porque no requiere de un aprendizaje previo, que puede resultar especialmente tedioso para profesionales no familiarizados con la ingeniería del software.

Sobre los atributos de la plantilla se ha visto que a) el atributo alcance es posible especificarlo dentro de la plantilla sin necesidad de usar otra, aumentando su legibilidad; b) el atributo valor objetivo no siempre es fácil de determinar. En los indicadores PPI-Virus, PPI-RBL y PPI-Spam el valor objetivo es 100% en el sentido de que el SIC desea detectar todos los virus. El problema es que se desconoce el número de virus real recibidos y se hace dificil determinar si se ha llegado al 100%.

Por último, desarrollar el modelo de procesos con los PPIs ha permitido identificar nuevos PPIs como el número de veces que se supera la cuota de disco, que permitiría estimar cuándo es necesario ampliarla, o bien, el porcentaje de *email* que proceden de fuente no autenticada, que sobrecargan el proceso con comprobaciones extras y cuya tendencia al alza interesa controlar.

4 Conclusiones y trabajo futuro

En este trabajo se ha presentado como escenario de aplicación el PN de recepción de correo desde una cuenta externa a la US a algún miembro de la comunidad universitaria para revisar la notación PPINOT y analizar su expresividad y su facilidad de uso en el ámbito de los PN híbridos.

Según nuestra experiencia, la representación gráfica de indicadores facilita comprender qué es lo que se quiere medir en cada uno de los procesos, así como la forma en que debe llevarse a cabo dicha medición. Por otra parte, ha permitido representar la totalidad de los PPIs que define el SIC en su panel de indicadores.

Tras describir los PPIs con PPINOT no hemos detectado especial problema al ser un proceso híbrido. Se han identificado algunas posibles mejoras como la necesidad de aclarar en qué situaciones es indistinto el uso de *Count Measure* o *State Condition Measure* y algunas puntualizaciones necesarias en las plantillas que, en general, facilitan la descripción y compresión de los PPIs.

En cuanto a los trabajos futuros, nuestra primera intención es calcular de forma automática los PPIs especificados. Trabajos previos [6], calculan dichos valores a partir del archivo de log generado durante la ejecución del BPMS *open source Activiti* [11]. En el caso del SIC, el elemento de partida son los archivos de logs que generan los servidores de correo.

Agradecimientos

Este trabajo ha sido parcialmente financiado por fondos FEDER, los proyectos de investigación TAPAS (TIN2012--32273) del Plan Nacional I+D+i, y COPAS (P12--TIC--1867) y THEOS (TIC--5906) del Plan Andaluz de Investigación.

A Carmen López y Javier de Miguel del SIC, por dejarnos "entrar en su casa" para estudiar el proceso y por su amable disposición para facilitarnos toda la información requerida.

Referencias

1. Del Río-Ortega, A. 2012. "On the Definition and Analysis of Process Performance Indicators." PhD Thesis, University of Seville.
2. Weske, M. 2012. Business Process Management: Concepts, Languages, Architectures. Berlin: Springer.
3. Grosskopf, A., G. Decker, and M. Weske. 2009. The Process: Business Process Modeling Using BPMN. Tampa, FL: Megan–Kiffer Press.
4. del Río-Ortega, A., M. Resinas, and A. Ruiz-Cortés. 2010. "Defining Process Performance Indicators: An Ontological Approach." In Proceedings of the 18th International Conference on Cooperative Information Systems (CoopIS). OTM 2010, Part I, 555–572. Berlin: Springer.
5. Del Río-Ortega, C. Cabanillas A., M. Resinas, and A. Ruiz-Cortés. 2012. "Ppinot: A Tool for the Definition and Analysis of Process Performance Indicators" In VIII Jornadas de Ciencias e Ingeniería de Servicios.
6. Del Río-Ortega, A., M. Resinas, C. Cabanillas, and A. Ruiz-Cortés. 2012. "On the Definition and Design-time Analysis of Process Performance Indicators." Information Systems 38 (4): 470–490.
7. Chase, G., A. Rosenberg, R. Omar, J. Taylor, and M. Rosing. 2011. Applying Real-World BPM in an SAP Environment. New York: SAP Press/Galileo Press.
8. Del Río-Ortega, A., M. Resinas, A. Durán, and A. Ruiz-Cortés. 2014. "Using templates and linguistic patterns to define process performance indicators" Enterprise Information Systems.
9. Durán, A., B. Bernárdez, A. Ruiz-Cortés, and M. Toro. 1999. "A Requirements Elicitation Approach based in Templates and Patterns". In Proceedings of Workshop on Requirements Engineering (WER), 17–29.
10. Van der Aalst, W.M.P., A.H.M. ter Hofstede, and M. Weske. 2003. "Business Process Management: A Survey". Springer-Verlag Berlin Heidelberg.
11. http://activiti.org/. BPMS Activiti. Accedido Junio 2014.
12. Shahin, A., and M.A. Mahbod. 2007. "Prioritization of Key Performance Indicators: An Integration of Analytical Hierarchy Process and Goal Setting." International Journal of Productivity and Performance Management 56: 226–240.
13. Decker, G., H. Overdick, and M. Weske. 2008. "Oryx – An Open Modeling Platform for the BPM Community." In Proceedings of the 6th International Conference on Business Process Management (BPM), 382–385. Berlin: Springer.
14. http://www.isa.us.es/ppinot/. Documentación PPINOT. Accedido Abril 2014.
15. http://labs.isa.us.es:8080/prspectives/index.html. Editor gráfico PPINOT. Abril 2014.

Data-Oriented Declarative Language for Optimizing Business Processes*

Luisa Parody and María Teresa Gómez-López and Rafael M. Gasca

Universidad de Sevilla, Sevilla, Spain,
{lparody,maytegomez,gasca}@us.es

Summary of the Contribution

Business process modelling constitutes an essential and crucial task in the Business Process Management. Typically, business processes, henceforth referred to as BP, are specified in an imperative manner, which define exactly how things have to be performed. But sometimes, a BP may be exposed to different environments and subjected to many conditions in which not always a sequence of activities can be described at design time. This is the reason why several authors have proposed languages to define BP as declarative models. These declarative languages tend to be used to describe the possible execution order of the activities, allowed or prohibited, instead of the exact order of the activities.

There are a significant number of researches that detect the necessity to include the data description into the BP model. Unfortunately this effort has only been applied to imperative models, not being the declarative models the focus of the studies, more centred on the order of activities. The role of data in declarative languages has not been very relevant, mostly limited to describe the execution or not of an activity, depending on the value of a variable of the dataflow. Unfortunately, none of them is worried about a declarative description of exchanged data between the activities, and how they can influence the model.

In this work, an analysis of the declarative languages found in the literature has been made in great depth. The analysis includes an study of how the most important declarative languages address data management, by means of the formalism for reasoning that they used (Linear Temporal Logic, Event Calculus,...); the capacity to include the data perspective; and the use of the declarative languages (validation, construction and/or assistance). Thanks to this analysis, the necessity to define a new language where the data aspects take more relevant place is demonstrated. In order to solve this lack in declarative languages, we propose a Data-Oriented Optimization LanguagE, called DOODLE, that represents graphically a declarative model which includes the BP requirements referring to data description. This new point of view of declarative languages focused on data permits to represent declaratively, the model of a business process according to

* This work has been published in the 22nd International Conference on Information Systems Development (ISD 2013), ranked as A in ERA and CORE Conference Rankings. This work has been partially funded by the Ministry of Science and Technology of Spain (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

data dependencies between the activities, and the objective function required to optimize the product of the BP in accordance to customer requirements. Compared with declarative languages, our proposal DOODLE includes data-oriented aspects in a declarative manner. It is done by means of a set of constraints that describe the data exchanged among the activities, when their relations cannot be defined explicitly at design time. Finally, DOODLE is applied to represent a motivating example: an on-line book store.

Apoyo a la Toma de Decisiones en la Compra de IaaS*

Octavio Martín-Díaz, Pablo Fernández, and Antonio Ruiz-Cortés

Dpto. Lenguajes y Sistemas Informáticos
ETS. Ingeniería Informática – Universidad de Sevilla
41012 Sevilla, España – Spain
omartindiaz@us.es, pablofm@us.es, aruiz@us.es

Resumen. La dificultad para decidir la compra de un IaaS (*Infrastructure as a Service*) depende de la complejidad de las opciones de compra dadas por su proveedor y de la complejidad del plan del cliente que quiere realizarla. Es habitual que estos tipos de servicios ofrezcan muchas configuraciones de uso diferentes, y para cada una de ellas sea posible disponer de varias opciones de compra. De este modo, decidir la mejor compra se convierte en una tarea que consume mucho tiempo, tediosa y propensa a errores. En este trabajo inicial, caracterizamos el problema con un caso de estudio ilustrativo y presentamos los desafíos inmediatos para mejorar las herramientas de soporte actualmente disponibles.

Palabras claves: IaaS, Opciones de Compra, SLA

1. Introducción y Motivación

La compra de un IaaS permite adquirir recursos de procesamiento y almacenamiento como alternativa a la adquisición de una infraestructura informática propia. Esta alternativa permite abaratar los costes asociados a la creación de dichas infraestructuras y su posterior renovación. El escenario de compra típico puede ser el siguiente: (1) un proveedor oferta las *opciones de compra* para un servicio, las cuáles suelen estar fijadas para una configuración determinada del servicio y un tiempo de uso; y (2) un cliente presenta su *plan de uso* y selecciona a partir de las opciones de compra un *plan de compras* que le sea rentable.

La dificultad para obtener un plan de compra depende de la complejidad de las opciones de compra y también de la complejidad del plan de uso del cliente que quiere hacer dicha compra. Las opciones de compra de un IaaS por parte de un proveedor pueden estar ligadas a servicios altamente configurables, de manera que la lista de precios puede llegar a ser abrumadora.

* Este trabajo ha sido cofinanciado por el Gobierno de España por el proyecto TAPAS (TIN2012-32273) y la Red Temática Científico-Tecnológica en Ciencias de los Servicios (TIN2011-15497-E), y la Junta de Andalucía por los proyectos THEOS (TIC-5906) y COPAS (P12-TIC-1867).

Por ejemplo, Amazon EC2 ofrece actualmente 7954 configuraciones de instancia diferentes [5]. Además, tanto configuraciones como opciones de compra pueden estar sujetas a variabilidad, es decir, el proveedor puede añadir o quitar configuraciones, añadir o quitar parámetros de las propias configuraciones, añadir o quitar opciones de compra, cambiar precios, o incluso cambiar el propio modelo subyacente para las opciones de compra, por ejemplo, modificando el método para calcular los precios.

Respecto al cliente, los planes de uso pueden referirse a uno o más servicios, del mismo o diferente tipo, con la misma o diferente configuración. Por ejemplo, un entorno de virtualización de prácticas de Ingeniería del Software tiene que atender a diferentes asignaturas, con diferente software, diferentes grupos y horarios, que pueden estar solapados.

Teniendo en cuenta las opciones de compra y un plan de usuario, el objetivo es encontrar un plan de compra óptimo, consistente en conocer cuántas instancias de cada opción de compra hay que reservar para ejecutar el software requerido sobre el IaaS.

En la actualidad existen herramientas que permiten la búsqueda óptima de un tipo de instancia entre varios proveedores [4], o varios tipos de instancia independientes [3]. Sin embargo, la toma de decisión sobre opciones de compra en escenarios complejos, tal y como describimos, sigue siendo un proceso manual que consume tiempo, es tedioso y propenso a errores. Así pues, se hace necesario un conjunto de técnicas y herramientas que soporte de manera automatizada el proceso de compra.

En este trabajo inicial, caracterizamos el problema y argumentamos los desafíos inmediatos, poniendo de relieve los parámetros clave mediante un caso de estudio con opciones de compra altamente configurables y planes de uso complejos.

El resto del trabajo se organiza como sigue. La sección 2 detalla un caso de estudio ilustrativo del escenario para caracterizar el problema. A continuación, la sección 3 presenta los desafíos para abordar el problema y la sección 4 el trabajo relacionado. Finalmente, la sección 5 expone nuestras principales conclusiones y trabajo futuro.

2. Caso de Estudio

En esta sección presentamos un ejemplo ilustrativo para identificar los parámetros clave en el escenario de compra de IaaS: servicios altamente configurables y complejidad de los planes de uso.

2.1. Opciones de Compra en Amazon EC2

Actualmente Amazon EC2 es uno de los IaaS más populares. Uno de sus elementos clave es la *instancia*, que representa un recurso virtualizado de procesamiento y memoria sobre el que se despliega una máquina virtual. La configuración de una instancia incluye un conjunto de parámetros muy variado.

Entre otros parámetros:

- Nombre identificativo de la instancia, por ejemplo, "m3.medium" o "m3.large".
- Uso previsto de la instancia: *light*, *medium* o *heavy*.
- Parámetros de procesamiento/memoria: número de CPU, 32 ó 64 bits, RAM o almacenamiento SSD/HD.
- Volumen de entrada y salida de datos: *very low*, *low*, *moderate*, *high* o *10-gigabit*.
- Duración de 1 ó 3 años.
- Sistema operativo: Linux, Windows, SQL-Windows o SQL-Web-Windows.

Cada configuración de instancia tiene sus opciones de compra (*purchasing options*) cuyos precios podrán variar según la zona en la que se encuentre desplegada la infraestructura real que le da soporte. Destacamos principalmente:

- *Bajo demanda* Se paga cuando se usa. Es la opción más cara, solamente recomendada para un uso esporádico de la instancia.
- *Reserva* Se paga por adelantado una parte del precio (*up-front*) a cambio de disminuir el pago por hora y garantizar la disponibilidad de la instancia. Hay tres opciones, a 1 y 3 años:
  - Ligero (*light*) Opción recomendada cuando se prevee que la instancia tenga un uso no intensivo, i.e. pocas horas cada día. El precio es muy similar (o igual en algunos casos) al uso bajo demanda, pero la disponibilidad de la instancia esta garantizada. Se cobran las horas de uso efectivo.

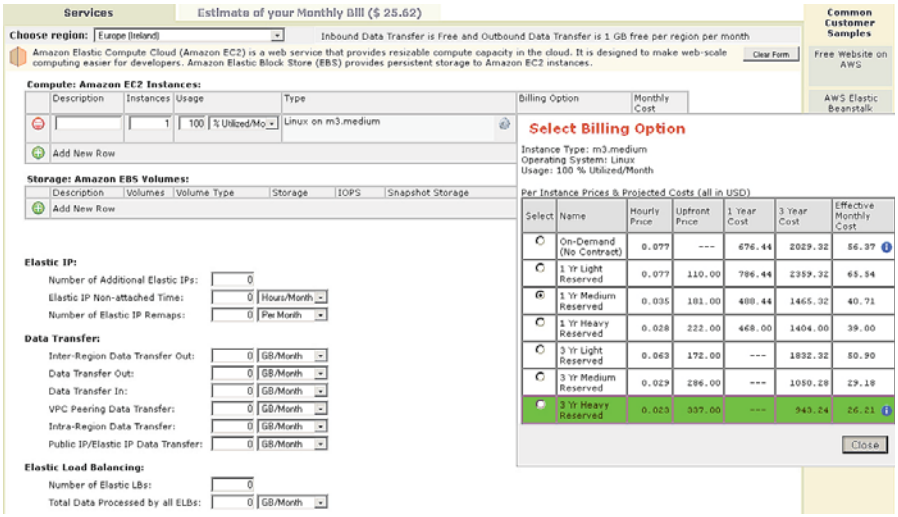

Figura 1. Calculadora de Amazon

- Medio (*medium*) Opción recomendada cuando se prevee un uso más continuado respecto a la anterior, de manera que el pago por adelantado es mayor y el precio/hora menor. Se cobran las horas de uso efectivo.
- Intenso (*heavy*) Opción recomendada cuando se prevee un uso casi continuo de la instancia. Adelanto de mayor cuantía, pero al precio/hora más bajo. En este caso se cobran todas las horas que cubren el período reservado, se haya hecho uso efectivo o no.

Dado que el precio de cada opción de compra puede variar en función de la configuración elegida, el número de instancias y la intensidad prevista de uso, encontrar el coste del servicio se convierte en una actividad tediosa y propensa a errores. Más si cabe cuando se agregan diferentes tipos de instancia u otras opciones que (por simplicidad) no hemos tenido en cuenta en este trabajo, entre ellas parámetros sobre volúmenes y transferencia de datos, balanceo de carga o IP elásticas.

Amazon ofrece una calculadora web para obtener dicho coste e incluso indicar la opción de compra recomendada. Por ejemplo, la Fig. 1 muestra su aspecto para una instancia de Amazon EC2 "*m3.medium*" de propósito general, con 1 CPU virtual, equivalente a 3 unidades de computación EC2, 3.75Gb RAM, 4Gb SSD, con Linux, en la Unión Europea[1]. Nótese que, para reservas de 1 año de duración, el porcentaje de uso debe ser superior al 19 % (unas 1650 horas anuales) para que la reserva de uso medio sea más rentable que la opción ligera, y superior al 80 % (unas 7000 horas anuales) para que la reserva de uso intenso sea la más rentable.

2.2. Planes de Uso de un Entorno de Virtualización de Prácticas

Presentamos un caso de estudio sobre *virtualización del ecosistema de herramientas de prácticas* en la docencia de Ingeniería del Software. Estas prácticas pueden necesitar una evolución muy dinámica desde dos puntos de vista. Desde el primero, las prácticas suelen utilizar software que requiere de recursos crecientes, siendo más que probable que queden obsoletos en un relativo corto período de tiempo. Por el segundo, el número variable de estudiantes realizando prácticas a lo largo de un curso lectivo. Por ello, la virtualización hecha posible por la compra de una IaaS ofrece perspectivas tanto de flexibilidad como de ahorro frente a dicha necesidad.

En Amazon EC2, la virtualización se consigue mediante la reserva de instancias, cuyas configuraciones sean adecuadas, sobre las que se despliega el ecosistema. Para ello, el profesor de una asignatura debe indicar:

- La configuración mínima necesaria de la instancia para ejecutar el ecosistema.
- La planificación de prácticas a lo largo del curso, que incluye fundamentalmente:

[1] `http://calculator.s3.amazonaws.com/index.html`

- El horario semanal de clase de los grupos de prácticas, así como el número de alumnos matriculados en cada grupo.
- Las fechas y horas de exámenes prácticos.
- Las aulas de libre acceso, proporcionando el número estimado de estudiantes que quieran utilizarlas y los períodos de mayor demanda.

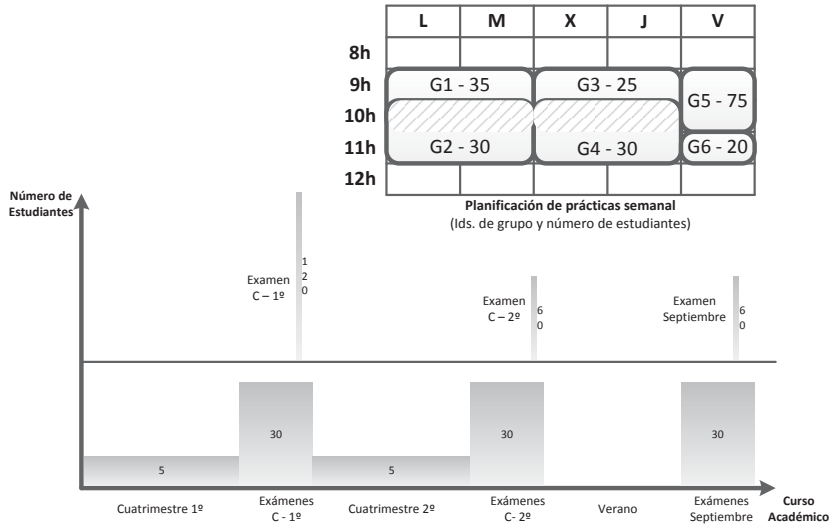

Figura 2. Ejemplo de planificación de prácticas

A modo de ejemplo, véase la planificación de la Fig. 2. Nótese que el calendario lectivo del curso académico consta de dos períodos cuatrimestrales, donde se imparten clases prácticas con una periodicidad semanal:

- Hay 4 grupos que reparten sus 4 horas semanales en dos días, existiendo solapes en sus horarios; más otros 2 grupos con menor número de horas. Estos grupos tienen matriculados a 35, 30, 25, 30, 20 y 75 estudiantes, respectivamente, para totalizar 215 estudiantes.
- Hay 3 períodos de exámenes, con un día previsto para las pruebas prácticas de la asignatura. Se prevee que haya una asistencia de 120 estudiantes en la convocatoria del primer cuatrimestre y 60 en la convocatoria del segundo cuatrimestre o en septiembre. Todas ellas tienen una duración de 4 horas.
- El aula abierta tiene prevista una asistencia de 5 estudiantes cualquier día durante el curso lectivo por 10 horas diarias. En períodos de exámenes y dos semanas previas, la asistencia puede ascender hasta 30 estudiantes.
- No hay docencia durante el período de vacaciones de verano[2].

[2] Por simplicidad, se han obviado otros tipos de descanso vacacionales o días festivos.

2.3. Obtención del Plan de Compras

Una vez que conocemos las opciones de compra que oferta Amazon EC2 y el plan de uso del profesor de una asignatura, es el momento de determinar el plan de compras, que consiste en obtener el número y tipo de instancias a reservar, que sea la opción más económica.

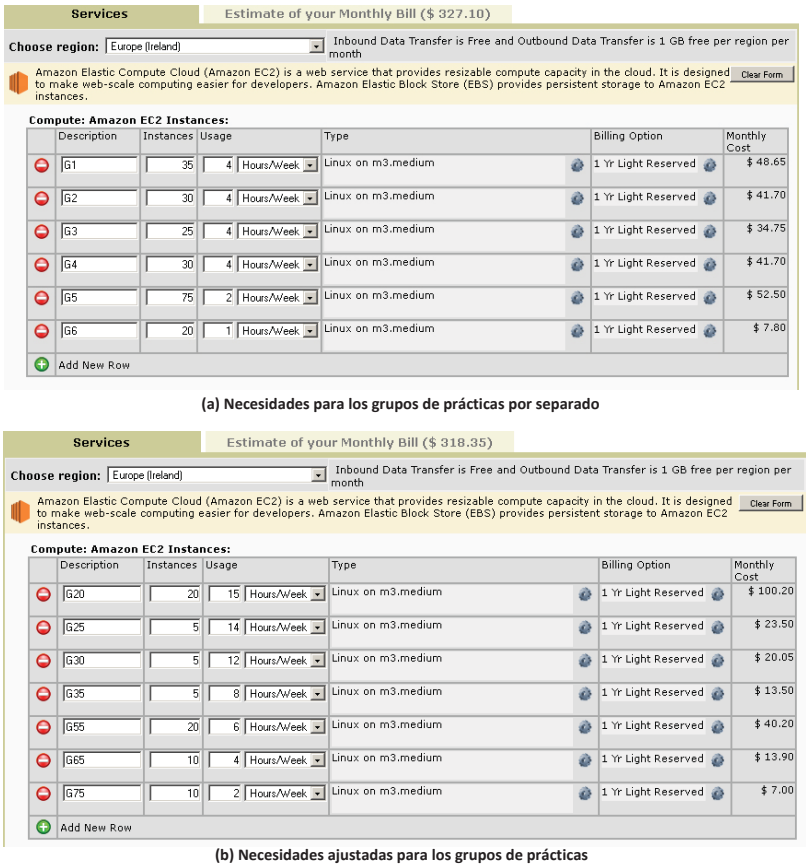

Figura 3. Comparativa de Estrategias de Compra

A modo de ejemplo y por simplicidad, tengamos en cuenta solamente los 6 grupos de estudiantes. El plan de compra más simple puede consistir en reservar una instancia de uso ligero para cada estudiante, es decir, para el G1 con 35 estudiantes, se reservan 35 instancias, y así sucesivamente. Entonces, la calculadora de Amazon nos da un coste de 327,10$ mensuales, como aparece en la Fig. 3(a), con la configuración ya vista anteriormente.

Si estudiamos en detalle el plan de uso, podemos encontrar planes de compra más eficientes y económicos. Por ejemplo, teniendo en cuenta los solapes en los horarios y que no siempre todos los estudiantes están haciendo prácticas a la vez. La Fig. 4 ilustra el método para conocer cuántos estudiantes simultáneos hay, durante cuántas horas, a lo largo de la semana.

La Fig 3(b) muestra el plan de compra resultante, cuyo coste baja a 318,35$ mensuales. Este plan de compra más ajustado se interpreta como sigue: entre los diferentes grupos de estudiantes, hay 20 estudiantes haciendo prácticas simultáneamente durante 15 horas, otros 5 más haciendo prácticas durante 14 horas, etc. hasta que llegamos a un ultimo grupo de 10 estudiantes más durante 2 horas, que cubre el pico de estudiantes que queda en el G5.

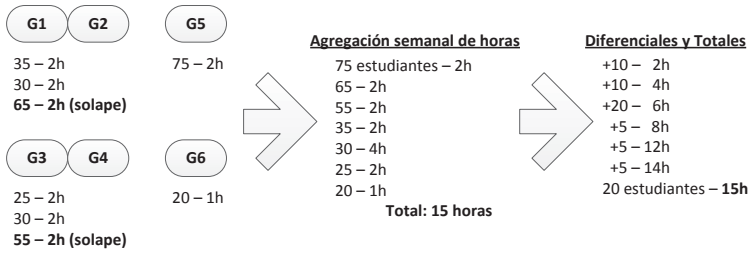

Figura 4. Cómputo de horas totales en el plan de uso

En este escenario, cabe preguntarse si acumulando horas de los planes de uso de diferentes asignaturas, aunque con necesidades de configuración similares, la reserva de instancias de uso medio o intensa llega a ser más económica. Estos cálculos son difíciles y necesitan ser automatizados, pero hasta lo que sabemos, Amazon no ofrece API de su calculadora y tampoco soporta el coste de varios grupos de instancias multiplexables en el tiempo.

3. Desafíos

La toma de decisión para seleccionar la mejor opción de compra en escenarios con servicios altamente configurables y planes de uso complejos ofrece un conjunto de desafíos que plantean un camino de investigación prometedor:

- Automatizar el proceso de compra a partir de documentos de acuerdos de nivel de servicio. Por ejemplo, se puede utilizar WS-Agreement [1], que es el estándar *de facto* para especificar acuerdos de nivel de servicio en la actualidad. Estos acuerdos podrían ser utilizados como documentos que contienen la información necesaria para expresar las opciones de compra (mediante plantillas) o los planes de uso (mediante ofertas de acuerdo).

- Mayor complejidad en los planes de uso. El soporte para expresar el plan de uso que ofrece Amazon u otros proveedores es limitado para algunos escenarios, lo que lleva a:
  - Que sean necesarios algunos cálculos extra para calcular el porcentaje de uso, debido por ejemplo a que los períodos se expresen en el mejor de los casos indicando el numero de horas por día: de lunes a viernes de 9 a 15 horas, excepto los viernes, hasta las 14 horas. Los tiempos de uso debieran estar por tanto en términos del dominio del problema.
  - Que en ocasiones, lo que es más grave, se opte por un plan de compras que no es el óptimo, por ejemplo cuando no se tiene en cuenta la posibilidad de que los grupos compartan instancias del servicio cuando ello es posible. Los grupos de instancias deben ser multiplexables en el tiempo, es decir, hay que tener en cuenta los solapes y que el tamaño de los grupos para un mismo tipo de instancia sea variable en el tiempo.
  - Que los tipos de instancia puedan ser homogeneos o heterogéneos, es decir, que se necesitan instancias con diferentes configuraciones.

  Las cuestiones relativas a la temporalidad ya han sido estudiadas en el contexto de WS-Agreement en [10].
- La Fig. 5 muestra la familia de *funciones de puntos de equilibrio* asociadas a cada opción de compra para 1 año correspondiente a la configuración "*m3.medium*", presentada en la sección 2. Nótese que se ha destacado la envolvente en la familia de curvas que determina el coste mínimo. El análisis de las envolventes muestra los umbrales sobre porcentajes de uso que determinan las opciones de compra más económicas, de manera que se puede interpretar el cálculo de coste más económico como el valor de la envolvente para el porcentaje de uso indicado.

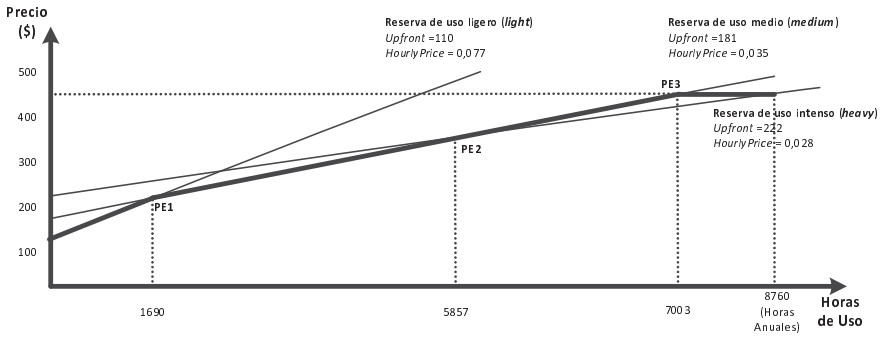

Figura 5. Familia de Puntos de Equilibrio

Esta interpretación facilitará en gran medida la automatización del cálculo del coste y se protegerá de las variaciones de las diferentes políticas de compra siempre que sea posible obtener automáticamente dicha envolvente a partir de la información disponible.

- Protección frente a las variaciones de las opciones de compra ofertadas por los proveedores. Además, diferentes proveedores pueden ofertar con diferentes modelos de compra; así mismo, un mismo proveedor puede modificar tanto el modelo como las opciones a lo largo del tiempo.
- Considerar otros modelos de compra. Las ofertas de los proveedores de IaaS actuales incluyen otros tipos ofertas dinámicas; por ejemplo, en Amazon EC2 además de las ofertas comentadas en nuestro caso de estudio, existen otros dos mercados adicionales: el mercado secundarios de instancias reservadas y el mercado de instancias de uso puntual (*Spot instances*).

4. Trabajo Relacionado

Hasta lo que conocemos, no hemos encontrado una propuesta que afronte el problema de la compra de servicios de IaaS altamente configurables con planes de uso complejos por parte del cliente. Pero sí que hay propuestas para solucionar partes más simples del mismo:

- En [5] se propone un método para encontrar la configuración óptima para una instancia, entre las miles de configuraciones que oferta Amazon EC2, en el contexto de migración de las aplicaciones software a la IaaS. En nuestro caso de estudio, este método puede utilizarse para encontrar la configuración mínima necesaria para el entorno de virtualización del ecosistema.
- En [4] se propone una herramienta que permite encontrar el mejor precio entre diferentes proveedores para una configuración dada. En nuestro caso de estudio, esta herramienta puede utilizarse para encontrar el proveedor óptimo para una configuración.
- En [3] se propone una herramienta similar, que permite encontrar el mejor precio entre diferentes proveedores para múltiples configuraciones de instancia independientes. Como la herramienta anterior, puede utilizarse en nuestro caso de estudio para encontrar el proveedor óptimo.

Entre las propuestas IaaS que tienen en cuenta el aspecto de la temporalidad en contexto similares al nuestro, encontramos:

- [2] propone el establecimiento automático de acuerdos de nivel de servicios, basado en WS-POLICY, con un caso de estudio basado en Amazon EC2.
- [7] propone el despliegue óptimo en una IaaS de un conjunto de servicios, que tiene en cuenta la variación del número de usuarios y el patrón probabilístico de invocaciones a dichos servicios.
- [11] propone algunos esquemas de negociación en IaaS. Presenta un caso de estudio en Amazon EC2 y propone un modelo de preferencias basado en funciones de descuentos y sensibles a la temporalidad.
- [8] también propone el problema de la negociación de SLA en IaaS.
- [9] analiza el problema del pago excesivo en IaaS debido al uso intensivo de servicios durante un periodo temporal, bajo un esquema pre-pago de servicios.
- [6] afronta el problema de analizar el uso concreto de las licencias software en IaaS, dado el gran dinamismo que hay en este contexto.

5. Conclusiones y Trabajo Futuro

En este trabajo inicial hemos presentado una caracterización al problema la toma de decisión para seleccionar la mejor opción de compra en escenarios con servicios altamente configurables y planes de uso complejos. Como caso de estudio se ha analizado el servicio de Elastic Computing Cloud (EC2) proporcionado por Amazon para la virtualización de entornos de prácticas. A partir de este análisis se han identificado los parámetros clave del problema que puedan servir como base para el desarrollo de sistemas de apoyo a la toma de decisiones en este contexto, exponiéndolos como una serie de desafíos abiertos al futuro inmediato.

Referencias

[1] A. Andrieux, K. Czakowski, A. Dan, K. Keahey, H. Ludwig, T.Ñakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement) Version 1.1 draft 20, September 2006.

[2] M. Chhetri, Q. Bao Vo, and R. Kowalczyk. Policy-based Automation of SLA Establishment for Cloud Computing Services. In *12^{nd} Intl. Symp. on Cluster, Cloud, and Grid Computing*, pages 164–171, Ottawa, Canada, 2012.

[3] Cloudorado.com. Cloud Computing Price Comparison Engine. `http://www.cloudorado.com/`.

[4] CloudScreener.com. Cloud Computing Comparison and Evaluation. `http://www.cloudscreener.com/`.

[5] J. García Galán, O. Rana, P. Trinidad, and A. Ruiz-Cortés. Migrating to the Cloud: a Software Product Line based Analysis. In *3^{rd} Intl. Conf. on Cloud Computing and Service Science*, pages 416–426, Aachen, Germany, 2013.

[6] M. Kim, H. Chen, J. Munson, and H. Lei. Management-Based Licensing Discovery for the Cloud. In *10^{th} Intl. Conf. on Service-Oriented Computing*, pages 499–506, Shanghai, China, 2012.

[7] K. Konstanteli, T. Cucinotta, K. Psychas, and T. Varvarigou. Admission Control for Elastic Cloud Services. In *5^{th} Intl. Conf. on Cloud Computing*, pages 41–48, Honolulu, Hawaii, 2012.

[8] K. Lu, R. Yahyapour, E. Yaqub, and C. Kotsokalis. Structural Optimization of Reduced Ordered Binary Decision Diagrams for SLA Negotiation in IaaS of Cloud Computing. In *10^{th} Intl. Conf. on Service-Oriented Computing*, pages 268–282, Shanghai, China, 2012.

[9] L. Lu and E. Elmroth. A Time-Interval-Based Credit Rerservation Approach for Prepaid Composite Services in Cloud Environments. In *9^{th} European Conf. on Web Services*, pages 158–165, Lugano, Switzerland, 2011.

[10] C. Müller, O. Martín-Díaz, A. Ruiz-Cortés, M. Resinas, and P. Fernández. Improving Temporal–Awareness of WS–Agreement. In *5^{th} Intl. Conf. on Service-Oriented Computing*, volume 4749 of *LNCS*, pages 193–206, Vienna, Austria, 2007.

[11] K. Vengataraghavan and R. Sundarraj. On Integrating Time Preferences and Concession Models into Cloud Computing Negotiations. In *Intl. Symp. on Cloud and Services Computing*, pages 157–163, Mangalore, India, 2012.