Introducing a Mashup-Based Approach for Design-Time Compliance Checking in Business Processes*

Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés

Universidad de Sevilla, Spain
{cristinacabanillas, resinas, aruiz}@us.es

Abstract. Business process compliance tries to ensure the business processes used in an organization are designed and executed according to the rules that govern the company. However, the nature of rules (expressed in natural language) and the large amount of elements that can be involved in them make their materialization and automated checking quite difficult. That is why the existing support for compliance checking is generally restricted to specific kinds of rules (e.g. rules affecting the control flow of the process). In this paper, we introduce *compliance mashups*, and show how a mashup-based approach can help solve the problem of rule specification and checking at design time. Some advantages of such an approach are that: (i) any kind of rule can be specified, which implies that each user can specify a rule according to his/her interpretation of the rule; (ii) building the compliance mashup is transparent to the formalism(s) used to implement it, so different techniques can be used together; and (iv) mashup components or parts of them can be re-used. As an example we use this approach to build mashups to specify and check rules related to human resource management in business processes at design time.

Keywords: Business process compliance, rule specification, compliance mashup, natural language disambiguation, design-time compliance checking.

1 Introduction

Business process (BP) compliance consists of ensuring the BPs used in an organization are designed and executed according to the policies that govern the company. Policies can be decomposed into rules that introduce constraints relating to different aspects of the BPs, such as the execution order of the activities (i.e. control flow), the data accessed and managed, or the people (a.k.a *human resources* or just *resources*) that participate in the process.

* This work has been partially supported by the European Commission (FEDER), Spanish Government under the CICYT project SETI (TIN2009-07366); and projects THEOS (TIC-5906) and ISABEL (P07-TIC-2533) funded by the Andalusian Local Government.

M. Bajec and J. Eder (Eds.): CAiSE 2012 Workshops, LNBIP 112, pp. 337–350, 2012.

Compliance can be checked at different phases of the BP lifecycle [1], which results in two big compliance checking modalities [2]. The most proactive way to check compliance is Forward Compliance Checking (FCC), which can be divided into two sub-approaches: Design-Time Compliance Checking (DTCC) and Run-Time Compliance Checking (RTCC). DTCC is usually performed after BP modelling with the aim of ensuring that the process is compliant with the given rules before its execution, thus saving time and effort to business analysts. An example is tool OPAL developed by some researchers from IBM Research China, which uses model-checking techniques to automatically verify control flow-related rules in BP models [3]. Nevertheless, there is a bunch of proposals to deal with DTCC based on diverse techniques. A common feature of most of them is that they rely on annotated BP models to check compliance [4,5]. RTCC techniques check rules at run time, so if a rule is violated or some problematic situation arises while running the process we might be able to solve the problem on time to avoid ending in a non-compliant state. This requires some kind of software for business activity monitoring (BAM) in BP management systems (BPMS). European Project COMPAS worked in this direction [6]. Finally, Backward Compliance Checking (BCC) focuses on determining whether past instances of a BP were compliant with rules from information stored in history logs. The result of BCC helps stakeholders to be prepared for audits. ProM [7], a tool for process mining, contains plugins to perform this kind of compliance checking, mostly focused on control flow issues [8]. A summary of features that should be considered when developing a BP compliance management system (BPCMS) that covers all the phases of the BP life cycle was introduced in [9].

In this paper we focus on FCC, particularly on DTCC. In DTCC, rules are checked against BP models, which involves translating them into a formal language that can be automatically processed. This is not an easy task. It is widely known that there is an important gap between BPs and rules indeed [10]. Different compliance rule modelling languages and ways to insert policy-related information in BP models have been introduced in the last years [11, 3, 12, 13], but there are important problems that still remain partially unsolved. The own nature of rules is a problem itself, as policies are described in natural language, which may be ambiguous. Therefore, two organizations may implement the same rule in two slightly different ways, sometimes voluntarily (i.e. for "business policy"), other times by chance (i.e. due to an unconscious different interpretation or a misunderstanding of the rule). For instance, let us suppose we must fulfill the following rule in our organization:

Rule 1: *There is a segregation of duties between the creation of a hiring resolution proposal and its processing.*

Segregation of duties (SoD) is a well-known authorization constraint that aims at avoiding problems due to a conflict of interests in the execution of two activities. This is achieved by distributing the responsibility to more than one resource (e.g. roles, positions, persons). If we are developing a rule modelling language,

the SoD concept may be itself an element of the language that can be used to state rules such as Rule 1. However, by acting like that we are losing nuances that come from the human interpretation of natural language. For example, Rule 1 could lead to at least two different implementations:

- Strict. The business manager can assume that, besides selecting different roles for the two tasks, it is necessary to guarantee that two different persons undertake the activities in order to prevent the scenario in which the same person plays the two roles involved and, hence, may execute the two activities affected by the rule, which could result in a conflict of interests.
- Slight. However, we may find an organization in which people must indicate with which role perform every task and which does not care about the same person executing the two activities involved in a SoD as long as each activity is performed with a different role. In this case, the result of the SoD checking would be different from that of the previous implementation.

Each piece of our rule modelling language must have an associated semantics. So, the question is: which of the previous implementations is covered by our language? Both? Taking into account the whole potential interpretation variability of natural language would lead to complex languages, which in turn would derive in their hard understandability and use.

Another issue is how to deal with all the aspects of the BPs that can be affected by rules (e.g. control flow, data, resources, temporal constraints). The wide bunch of possibilities regarding BP aspects collaborate in making it difficult to develop such a language. Furthermore, some rules involve more than one aspect. This, together with the aforementioned interpretation problem, generate a complex and varied casuistry that makes it hard to define a *declarative* domain-specific language (DSL) expressive enough to address all kinds of rules. As a consequence, most of the techniques proposed so far limit their scope, e.g. to one or two BP aspects, giving rise to many ad-hoc approaches. The conclusions of a study we carried out on approaches dealing with BP compliance can be found in [14]. Besides, some techniques rely on one specification formalism for rule definition such as [13]. However, one formalism usually allows only some types of checks, so the entire casuistry is not covered like that. The mixture of different formalisms would be necessary.

Finally, parts of some rules can be re-used in the definition of other rules. Considering this in our rule modelling language would save effort to business analysts or to the person in charge of modelling the rules of a company (e.g. a compliance expert). Some pattern-based approaches such as BPMN-Q [12] kind of cover this issue. However, the problem of the large casuistry is still latent.

We propose *mashups* as a mechanism to provide an *operative* specification of rules and check design-time BP compliance. In this paper we will reveal how this approach allows us to address the aforementioned challenges and overcome the aforementioned issues. Mashups are easy to understand and use [15], and they can be implemented in many different ways, e.g. by using common spreadsheets [16]. Applied to policies, mashups let us handle different interpretations

of natural language by re-combining the mashup components used to specify a rule, thus dealing with natural language ambiguity.

Furthermore, different formalisms can be used together in a single mashup, provided that we have a real way to connect the information resulting from a technique with the input of another approach[1]. Therefore, the specification of rules by an end user may be independent of the underlying formalism for compliance checking. Besides, mashups offer, among other interesting features, flexibility, portability and re-usability, so rules already defined (or part of them) can be saved and used later to build other mashups. The fact that they have already been used for analysis purposes in other domains [17, 18], motivated us to explore their applicability to check BP compliance rules.

Finally, note that, although the focus of this paper is on DTCC, the idea behind our approach can be applied to other phases of the BP lifecycle and, as a matter of fact, we are currently working on extending the approach to RTCC and BCC.

The rest of the paper is structured as follows. In Section 2 we introduce mashups and their main types of components, accompanied by a generic example. Section 3 contains an explanation of our proposal, which is exemplified by applying it to a specific kind of compliance rules in Section 4. Finally, some conclusions and a summary of ongoing and future work are presented in Section 5. To improve the understandability of the explanations, and due to space limitation, references to related work are given throughout the paper.

2 Introduction to Mashups

Mashups are a hot concept in IT nowadays. A mashup is a data-driven workflow (a.k.a. *dataflow*) built with information from one or more data sources, and it is based on the re-use of contents and functionalities. Mashups were developed to build new Web services or applications from existing data in an "easy" way, so the end user does not need to have specific technical knowledge, but only knowledge of the problem domain [15]. They have already been applied to address problems such as the analysis of molecular biology in bioinformatics [17] and the simplification of patient management in hospitals [18], and there are some mashup makers in the market such as Intel Mashup Maker, Yahoo! Pipes or IBM Mashup Center. The Google App Engine also gives support to the previous Google Mashup Editor. There is a large amount of mashup examples available on the Web, e.g. more than 6,000 mashups can be found at `http://www.programmableweb.com/`. With tools such as Yahoo! Pipes anybody can build and publish a mashup in the Internet.

To show mashup appearance and use we have created the mashup in Figure 1. It returns the last 25 international news of the New York Times and The Australian digital newspapers[2]. Any researcher could want to have a similar

[1] The complexity of a mashup is within each component, and the greatest effort is put on how to integrate them.

[2] It can be run at `http://pipes.yahoo.com/cabanillasmashups/worldnews`.

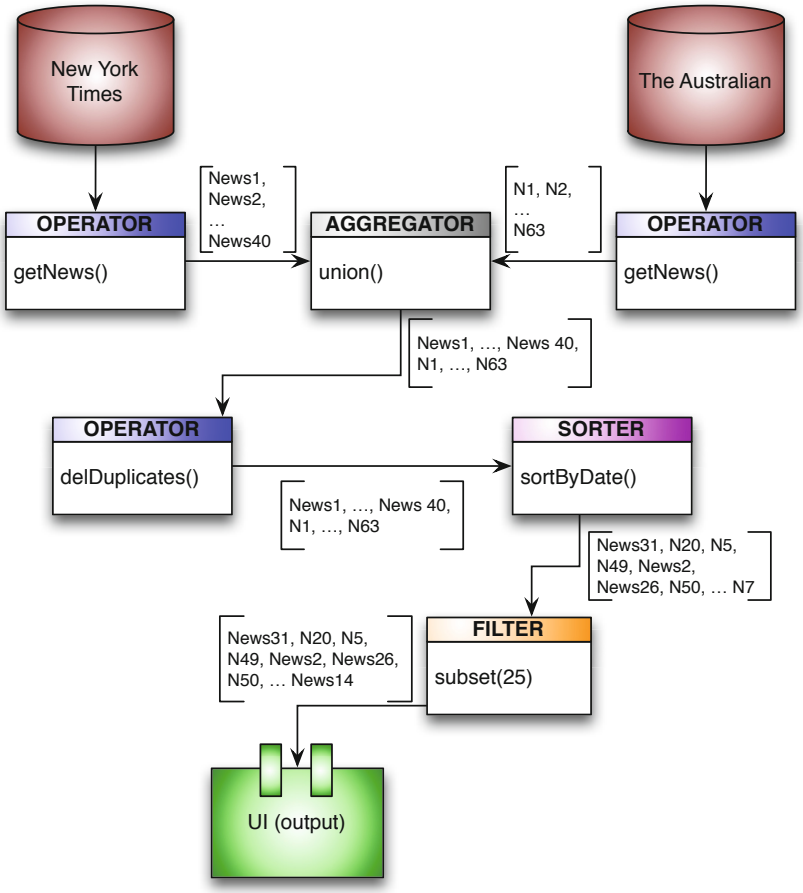


Fig. 1. Mashup collecting the last 25 world news from two digital newspapers

mashup to be kept up to date of his/her research interests, e.g. a mashup that automatically places on a map the venue cities where the next editions of his/her favourite conferences take place.

As illustrated in the figure, mashup editors allow the definition of the dataflow by connecting two main types of components: data sources and flow components.

Data sources range from data warehouses to URLs pointing at RSS feeds or any kind of accessible information.

Flow components are the elements in charge of operating on data, so they all have input and output. The input data they receive can come from another mashup component or from a data source, and the last component provides the information required to the output UI. Flow components can be generic-purpose components such as those that handle collections of elements to sort or join them, and domain-specific (DS) components, which implement functions specific to the problem domain, such as handling geographical location data to enrich Google Maps with external information. Some frequently used flow components include the following:

- *Filters.* They narrow down the flow of data, supporting the transformation of the information.
- *Aggregators.* They join or group data according to some criteria.
- *Operators.* They extract, elaborate and transform the information, constituting a very important part of the ETL (Extract-Transform-Load) process [19] data must undergo from the input of the mashup to the output UI. They range from operators that implement well-known functions such as *count*, *min*, *max* and *average* (i.e. general-purpose) to operators that handle strings or extract and builds geographical location information (i.e. domain-specific).
- *Sorters.* They return the same input data but in a specific order.

Languages for mashup representation, such as Enterprise Mashup Markup Language (EMML)[3], provide support to create and use at least the aforementioned components and they can usually be extended to include new DS components, if required. For insights about how to build high-quality mashups we recommend the reading of [20].

3 Mashups for BP Compliance Checking

We propose the use of mashups as *a language to provide an operative specification of rules and to query BP models.*

Definition 1. A *compliance mashup* is an operative DSL that allows the integration of heterogeneous data sources and the specification of compliance rules over subsets of the information that can be extracted from them.

In them, the data sources are the repositories where the organization stores the different models it uses, e.g. business processes, organizational structures, data and so on. Regarding the flow components of the mashup specification, a set of both general-purpose and DS components (filters, operators, sorters, etcetera) may be necessary to manipulate and transform the data coming from these models. In particular, in the case of DS operators, these components will encapsulate *analysis operations* (or queries) on models that enable the creator of the mashup to extract from the models the information he/she needs to check a compliance rule.

DS operators can be implemented using different analysis techniques. For instance, BPMN-Q is a language aimed at querying BP models regarding control flow and data [21] (e.g. it returns information on whether an activity is executed before or after another activity). There also exist mechanisms to analyse the data perspective of BP models, as long as these models have data-related information [22]. Other proposals deal with resource analysis in BP models with resource assignment information such as RAL or Business Activities. RAL is a DSL for the representation and analysis of resource assignments in BP models. The analysis

[3] `http://www.openmashup.org/omadocs/v1.0/index.html`

mechanism is provided by means of its ontology-based semantics [23]. Business Activities are a UML extension to integrate process flows and process-related RBAC models with resource-related constraints. The violation of constraints such as mutual exclusion between activities can be detected [24].

The set of available DS operators will depend on the kinds of checks we have to perform over the BPs of our organization. So, as aforementioned, some of them will implement mechanisms to check for some data-related functionality, some others will be focused on dealing with control-flow information, and so on.

Finally, the data sources and the flow components are connected in a dataflow to check for compliance rules.

As can be seen, building a mashup is not a hard task, assuming that all the logic within the components is implemented. They thus allow us to re-use existing solutions, avoiding to re-invent the wheel.

4 Applying Mashups to Resource-Related Compliance by Design

We are going to show how to build a mashup to specify and check rules related to resource management in BPs at design time. It is one of the aspects usually affected by rules nowadays, as we can find plenty of policies that regulate resource management in a company, to be named:

- SoD is well-known in financial accounting systems as a mechanism to prevent from fraud and error. In IS, it helps reduce the potential damage from one person's actions by disseminating the tasks and associated privileges for a specific BP among multiple users. A big part of the Sarbanes-Oxley Act (SOX)[4] is devoted to manage internal control in IT, in which SoD is a key concept.
- The Health Insurance Portability and Accountability Act (HIPAA)[5] pays special attention to who can do certain tasks in order to preserve the privacy of confidential information and to avoid fraudulent use of private data.
- Besides rules coming from well-known policies, each company has its own resource-related business rules that are defined ad-hoc to its BPs, e.g. to state what kind of resource (e.g. a role or a specific person) is in charge of each task.

As a use case we will use a real process designed and utilised in the Andalusian Institute of Public Administration (IAAP) that represents the procedure to create and process a resource resolution proposal for hiring people. This process has a high use frequency in the Andalusian Public Administration, which serves to more than 8 million end users. It has been modelled in BPMN for the ease of understanding (cf. Figure 2). As depicted in the model, once a draft of a resource resolution proposal is created, it is concurrently sent to the Consultative

[4] http://www.soxlaw.com/

[5] http://www.hhs.gov/ocr/privacy/

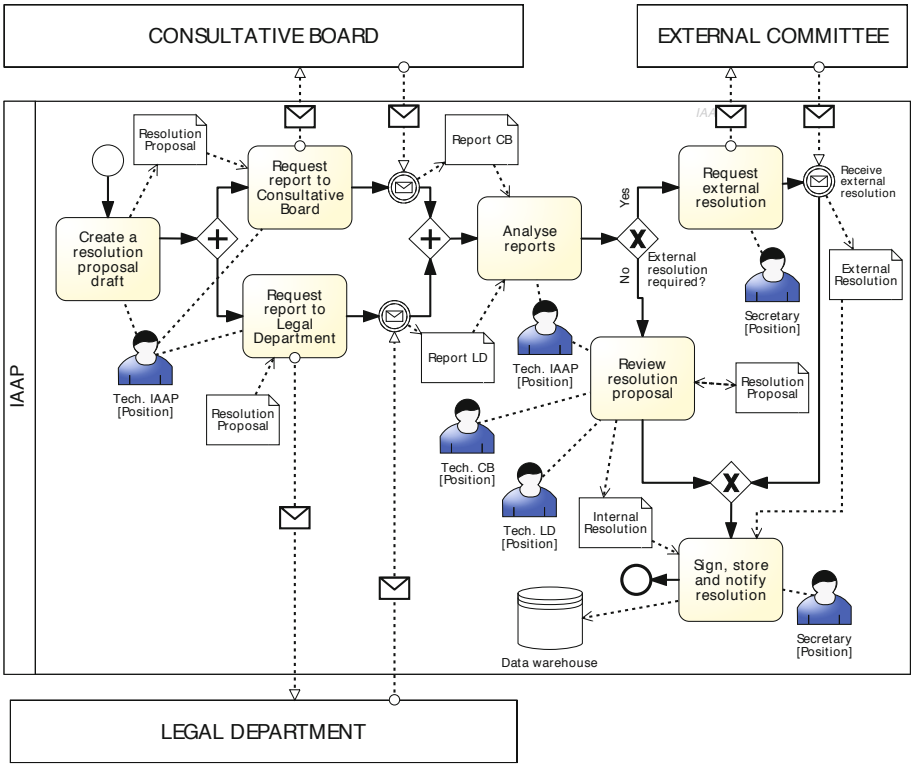
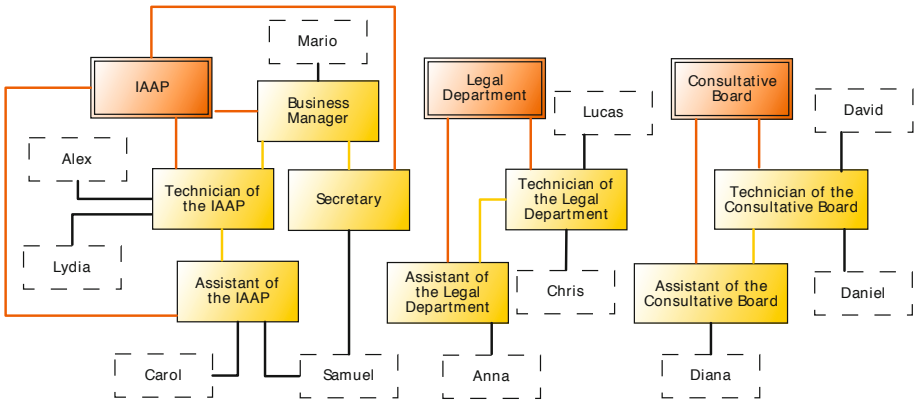


Fig. 2. Excerpt of the process to create and process a resource resolution proposal

Board and to the Legal Department for it to be evaluated. After receiving both evaluations the IAAP analyses them and decides whether an external resolution is required. In that case, a request is sent to an external committee, which must create and send a new resolution. Otherwise, the resolution proposal is reviewed and changes are applied to the initial draft. In any case, the documents generated are signed and archived, and the resolution result is appropriately notified.

Since we focus on resource-related compliance checking, we must be aware of the organizational structure of the IAAP. Figure 3 shows it regarding Administrative Resource Management. There are three organizational units called IAAP, Legal Department and Consultative Board, corresponding to different work teams. Eight positions (Business Manager, Technician of the IAAP, Assistant of the IAAP, Secretary, Assistant of the Legal Department, Technician of the Legal Department, Assistant of the Consultative Board, and Technician of the Consultative Board), occupied by a total of eleven people[6] (shown in white dash-lined rectangles in the figure), are associated to these units. Positions are connected to each other to represent hierarchical relations between them.

[6] Their names have been changed for privacy reasons.



Position	Role
Business Manager	Accountable
Technician of the IAAP	Technician Analyst
Technician of the Legal Department	
Technician of the Consultative Board	
Assistant of the IAAP	Technical assistant Document writer
Assistant of the Legal Department	
Assistant of the Consultative Board	
Secretary	Document signer

Fig. 3. Excerpt of the organizational structure of the IAAP

Figure 2 also depicts the resource assignments to the process activities. Note that assigning several resources to an activity means any of them can execute it, e.g. activity *Review resolution proposal* can be done by a Technician of the IAAP, a Technician of the Consultative Board or a Technician of the Legal Department.

Given this scenario we are interested in specifying and checking Rule 2:

Rule 2: *The creation of the resolution proposal draft and its revision after the evaluations of the Consultative Board and the Legal Department have to be performed by different roles.*

The Business Manager of the IAAP would be very interested in knowing if Rule 2 is met given the current BP model and the organizational structure in order to do the proper changes before running the process, if necessary. As we are checking the rule at design time, some considerations have to be done. Specifically, we should check that given the current resource assignments to activities, the same role can *never* execute the two mutual exclusive activities. Otherwise we can consider the process as non-compliant because we cannot ensure the rule is always fulfilled, that is, it could be met or violated depending on the specific resource allocation carried out at run time.

Figure 4 defines Rule 2 following the aforementioned consideration. In this compliance mashup we have two different data sources: (i) a repository of resource-aware BP models, i.e. models enriched with resource assignments; and

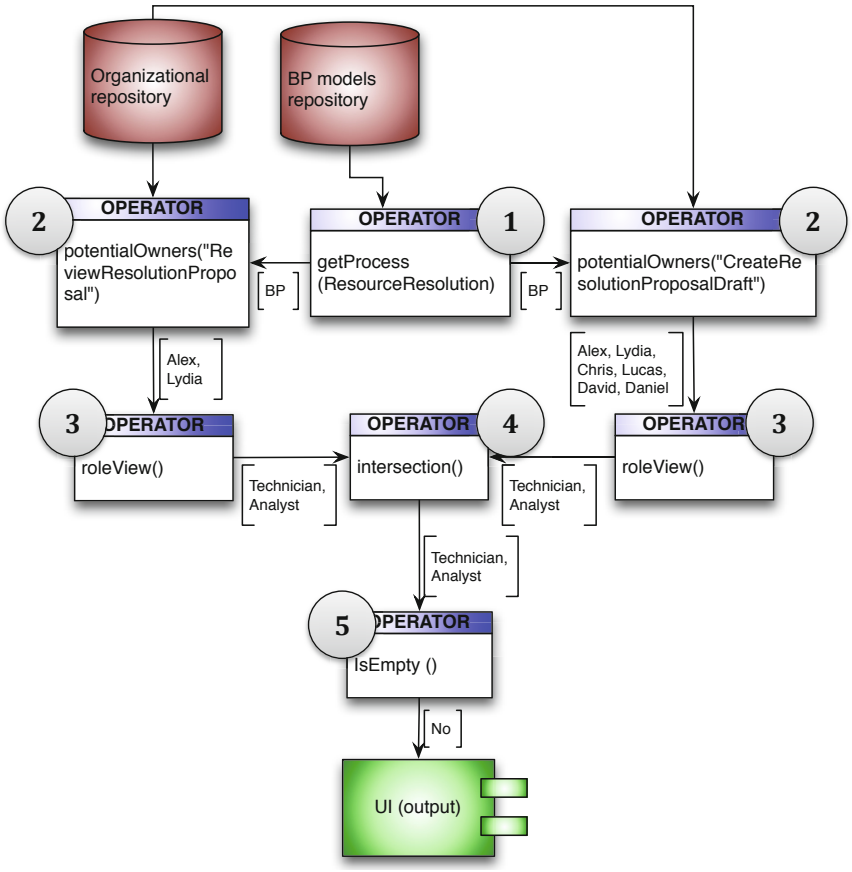


Fig. 4. Compliance mashup to define and check Rule 2

(ii) a repository of information related to the resources of the company (i.e. its organizational model, information about each resource and the like). The steps given to specify and check the rule are the following:

1. The BP affected by the rule is extracted from the repository of BP models.
2. The potential performers/owners (i.e., who *may* be allocated an activity at run time) of the two activities involved in Rule 2 are calculated.
3. As we are interested in roles, we generate an output on the basis of potential roles instead of potential persons, for which information from the organizational model is required (cf. Figure 3 to understand better).
4. Only the roles common to both sets of potential performers/owners must be selected.
5. If the resulting set is empty we can assure the rule is being always met. Otherwise we should state the rule may be violated at run time.

Therefore, in this concrete case the BP model is not compliant with Rule 2.

As shown, the mashup is easily built by selecting and connecting the proper components. Their inner implementation is not important to end users. Therefore, non-technical people from our organization could benefit from several mashup platforms that let any person with knowledge on the application domain create a mashup, with no need of technical knowledge. The functionality of the mashup could even be embedded in other existing applications, e.g. Kongdenfha et al. [16] have presented an approach that allows users to easily build mashups within a familiar spreadsheet environment.

If we wanted to give a slightly different meaning (or materialization) to Rule 2, we could re-configure the properties of the convenient components in Figure 4, reconnect them, or even insert new components to deal with the new interpretation (e.g. to force different people performing the two activities involved).

Besides, mashups can return different types of results (e.g. numeric values, text, boolean values). It means the output UI can show not only compliance checking results, but also the result of any *analysis operation* over BP models. For instance, we could build a mashup to obtain the list of activities that can be executed after a given activity. Therefore, we can use mashups for different kinds of analyses related to BP compliance and, in general, to BP management, always relying on the basis of re-using and integrating techniques. New approaches could be implemented and included as new components.

4.1 DS Components for Resource-Related Compliance Mashups

Most of the flow components depicted in Figure 4 are DS elements, specifically DS operators. Table 1 depicts some DS operators in the domain of resource-related compliance checking. Implementing some of them is quite trivial, e.g. those focused on the extraction of specific information from BP models or organizational models, as long as we have proper mechanisms to access the data. However, there are other DS operators whose implementation may not be so easy. That is the case of *potentialOwners*, in which we have to figure out what the candidates to execute the task at run time are from the resource assignments in the BP model. This involves an *analysis* of the BP model and the organizational model together to obtain the required information. Several approaches have been developed so far to perform this kind of analysis. For instance, RAL Solver, based on the aforementioned RAL, uses the HermiT DL reasoner to resolve RAL expressions and implement all of the operations depicted in Table 1. Similarly, many of those operations can also be implemented using the aforementioned Business Activities [25], or by means of the model-checking algorithms proposed by Wolter et al. to verify control access constraints in BPs [26]. Any of these approaches (or others) could be used to execute the mashup in Figure 4.

It is important to point out that the list of DS operators presented in Table 1 is not exhaustive. Indeed, operations to check properties related to dynamic issues in BPs are also necessary in order to consider the control flow of the process. Formalisms such as Petri nets [27] and BPMN-Q [12] have been widely studied and used in this kind of checks. Flow components performing functions for control-flow management could thus be incorporated to the mashup.

Table 1. DS operators for resource-related compliance checking

DS operation	Output	Function
Operations on business process models		
getProcess (PID)	BP	It returns the BP elements of the selected BP model
getActivities (PID)	[Activity]	It returns the set of activities of the selected BP model
Operations on organizational models		
getPerson (ID)	Person	They return the information associated to a specific element of an organizational model
getPosition (ID)	Position	
getRole(ID)	Role	
getUnit(ID)	Unit	
Operations using information from several data sources		
potentialOwners (A, OM)	[Person]	Potential performers of an activity according to its associated resource assignment expression. Sometimes looking through the organizational model may be required
potentialOwners ([Activity], OM)	[Activity: [Person]]	Potential performers of a set of activities. Sometimes looking through the organizational model may be required
positionView ([Activity: [Person]], OM)	[Activity: [Position]]	It classifies the persons by positions (according to the organization structure) and shows the positions associated with the activities
roleView ([Activity: [Person]], OM)	[Activity: [Role]]	This operation classifies the persons by roles (according to the organization structure) and shows the roles associated with the activities
unitView ([Activity: [Person]], OM)	[Activity: [Unit]]	It classifies the persons by organizational units (according to the organization structure) and shows the units associated with the activities

5 Conclusions and Future Work

The use of mashups to specify and check compliance rules provides a framework that may help to simplify an important problem in business process compliance management, which is the expression of compliance rules in a way that enables their automated checking. Since rules are defined by composing operators, mashups provide flexibility to give stricter or slighter meanings to them, as desired. Besides, mashups are quite easily understandable, which allows their creation or modification by people without technical skills. Furthermore, compositions can be saved as new mashup components, making them reusable.

This proposal provides a common approach to specify and check compliance rules regardless of the compliance domain provided as long as there is technical and technological support to implement the components for the specific domain. That is, we have applied mashups in DTCC, but with different data sources and DS flow components we could check other kinds of compliance. For example, building mashups for BCC would give rise to UIs that could work as a dashboard for the visualization of compliance issues for later audits [28].

Regarding the validation of our approach, we have just started a project with an IBEX-35 multinational company. The goal of the project is to develop a mashup-based system that allows the specification and checking of all the rules the company has to meet (basically rules from Internal Control Systems of Financial Information –SCIIF–, law decree L262, SOX, and some ad-hoc business rules). In the policies the company needs to check we have already found cases of ambiguous statement of rules in which a specific interpretation must be given, e.g. "database security parameters must be checked at least once a year". The following question emerged when modelling this rule: "a fiscal year or a year since the last time it was checked?" This reaffirms the need of providing flexibility for rule modelling.

Besides, the solution must cover all the compliance-related features described in [9], giving rise to a full-coverage BPCMS. Therefore, we are studying how to extend the approach presented in this paper to address more types of BP compliance (e.g. run-time issues). In this project, we also plan to evaluate how easy the building of compliance mashups actually is for non-technical people.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
2. Kharbili, M.E., de Medeiros, A.K.A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: MobIS 2008, pp. 107–113 (2008)
3. Liu, Y., Muller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal 46, 335–362 (2007)
4. Ghose, A.K., Koliadis, G.: Auditing Business Process Compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
5. Weber, I., Governatori, G., Hoffmann, J.: Approximate Compliance Checking for Annotated Process Models. In: Advances in Enterprise Engineering - GRCIS workshop at CAiSE 2008 (June 2008)
6. Silveira, P., Rodríguez, C., Birukou, A., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, Z.: Aiding compliance governance in service-based business processes. In: Non-Functional Properties for Service-Oriented Systems: Future Directions (NFPSLA-BOOK-2011). IGI Global (2011)
7. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: ProM: The Process Mining Toolkit. BPM (Demos) (2009)
8. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
9. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Exploring Features of a Full-Coverage Integrated Solution for Business Process Compliance. In: Salinesi, C., Pastor, O. (eds.) CAiSE Workshops 2011. LNBIP, vol. 83, pp. 218–227. Springer, Heidelberg (2011)
10. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. J. Vis. Lang. Comput. 22, 30–55 (2011)
11. Yu, J., Manh, T.P., Han, J., Jin, Y., Han, Y., Wang, J.: Pattern Based Property Specification and Verification for Service Composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)

12. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
13. Governatori, G., Sadiq, S.: The Journey to Business Process Compliance. In: Handbook of Research on BPM, pp. 426–454. IGI Global (2009)
14. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Hints on how to face business process compliance. In: III Taller de Procesos de Negocio e Ingeniería de Servicios (PNIS 2010) en JISBD 2010, pp. 26–32 (2010)
15. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12, 44–52 (2008)
16. Kongdenfha, W., Benatallah, B., Vayssière, J., Saint-Paul, R., Casati, F.: Rapid development of spreadsheet-based web mashups. In: WWW 2009, pp. 851–860 (2009)
17. Hull, D., Wolstencroft, K., Stevens, R., Goble, C.A., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. Nucleic Acids Research 34(Web-Server-Issue), 729–732 (2006)
18. Baresi, L., Guinea, S.: Mashups with Mashlight. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 711–712. Springer, Heidelberg (2010)
19. Akkaoui, Z.E., Zimanyi, E.: Defining ETL worfklows using BPMN and BPEL. In: Proceeding of the ACM Twelfth International Workshop on Data Warehousing and OLAP, Hong Kong, China, pp. 41–48. ACM (2009)
20. Cappiello, C., Daniel, F., Koschmider, A., Matera, M., Picozzi, M.: A Quality Model for Mashups. In: Auer, S., Díaz, O., Papadopoulos, G.A. (eds.) ICWE 2011. LNCS, vol. 6757, pp. 137–151. Springer, Heidelberg (2011)
21. Sakr, S., Awad, A.: A framework for querying graph-based business process models. In: WWW 2010, pp. 1297–1300 (2010)
22. Cabanillas, C., Resinas, M., Ruiz-Cortés, A., Awad, A.: Automatic Generation of a Data-Centered view of Business Processes. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 352–366. Springer, Heidelberg (2011)
23. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Defining and Analysing Resource Assignments in Business Processes with RAL. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 477–486. Springer, Heidelberg (2011)
24. Strembeck, M., Mendling, J.: Modeling process-related RBAC models with extended UML activity models. Inf. Softw. Technol. 53, 456–483 (2011)
25. Schefer, S., Strembeck, M., Mendling, J., Baumgrass, A.: Detecting and Resolving Conflicts of Mutual-Exclusion and Binding Constraints in a Business Process Context. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 329–346. Springer, Heidelberg (2011)
26. Wolter, C., Miseldine, P., Meinel, C.: Verification of Business Process Entailment Constraints Using SPIN. In: Massacci, F., Redwine Jr., S.T., Zannone, N. (eds.) ESSoS 2009. LNCS, vol. 5429, pp. 1–15. Springer, Heidelberg (2009)
27. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
28. Silveira, P., Rodriguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, Z.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: NFPSLAM-SOC 2009 (2009)