Distrib Parallel Databases (2016) 34:145–178
DOI 10.1007/s10619-014-7169-3

A view-based monitoring for usage control in web services

Hassina Meziane · Salima Benbernou · Mohand-Said Hacid · Zaki Malik · Mike Papazoglou

Published online: 14 December 2014

Abstract Quality of service (QoS) can be a critical element for achieving the business goals of a service provider, and accepting a service by the customer. The criticality is more pronounced when the service provider handles the non-functional QoS attribute of privacy, i.e., privacy related to the customer's personal data. In this regard, the customer needs some guarantee(s) from the service provider about confidentiality management, leading to overall quality characterization of the provided service. A service level agreement (SLA) is primarily intended to specify (in terms of clauses) the level of such non-functional QoS delivered to the customer. The aim is to provide customers with tools that show the fulfillment of QoS guarantees, through SLA monitoring process. In this paper, we address the problem of usage control of private data in service based applications ensuring end-to-end QoS capabilities. We

H. Meziane
University of Oran, Es Senia, Algeria
e-mail: meziane.hassina@univ-oran.dz

S. Benbernou (✉) · H. Meziane
Université Paris Descartes, Paris, France
e-mail: salima.benbernou@parisdescartes.fr

M.-S. Hacid
Université Claude Bernard Lyon 1, Villeurbanne, France
e-mail: mohand-said.haci@univ-lyon1.fr

Z. Malik
Wayne State University, Detroit, USA
e-mail: zaki@wayne.edu

M. Papazoglou
Tilburg University, Tilburg, The Netherlands
e-mail: mikep@uvt.nl

propose a query containment based approach to support the monitoring of privacy-aware SLA compliance, that spells out a customer's privacy rights, and shows how the customer's private information must be handled by a Web service provider. We introduce the private data usage flow model upon which the monitoring is performed to observe the data usage flow, and capture the privacy vulnerabilities that may lead to non-compliance. The model is built on top of (i) properties and time-related privacy requirements to be monitored, and (ii) a set of identified privacy violations. As proof of concept, a privacy aware SLA monitoring system, which is an easy-to-use, and efficient tool for observing the dynamic private data usage flow is developed. Experiment results indicate the relevance and applicability of the proposed approach.

Keywords Privacy aware SLA · Usage control · Monitoring · Usage flow view · Query containment · Compliance

1 Introduction

1.1 Motivation

In the last decade, the service oriented architecture (SOA) paradigm for Information Systems development has emerged as an interesting approach, allowing seamless integration among organizations that offer their expertise as Web-enabled software services (e.g., Web services). In the beginning, the interests of researchers and practitioners were focused on the functional aspects of these software services and associated description. Because of the increasing agreement on the implementation and management of the functional aspects of these services; such as the adoption of WSDL for service description, SOAP for Web service messaging, or WS-BPEL for Web service composition, the research interest is shifting towards the non-functional or quality aspects of Web-enabled software services, commonly referred to as Quality of Service (QoS). For a complete discussion on QoS aspects of service-oriented systems, please see our prior work in [27]. In SOAs, service providers are *a priori* unknown to the customers, i.e., they do not know in advance which service they are invoking, and service binding can be made dynamically according to the customer's functional and quality requirements. Due to this loose coupling between service customers and providers in a service-based application, the management and assurance of QoS aspects are becoming of utmost importance [8]. Moreover, the proliferation of Web services is accompanied by a huge amount of data (including personal data) exchanged by the interacting entities to complete the execution of a service. This has led to an increase in the number of inappropriate usage and leakage of personal data, and *privacy* has emerged as the foremost concern, and a challenging issue. For instance, most of us who have purchased something on the Internet have experienced the pause and wondered if it is "safe" to enter one's credit card information. Clearly, the more one is exposed to new services on the Internet, the more varied is the personal information demanded by these services, and the more one wonders whether this personal information entered would be kept safe. The major problem in this regard is determining the trustworthiness of Web-based applications in terms of controlling the private data usage (to provide more confidentiality). Such a need leads to building and managing service-based systems which provide desired end-to-end privacy awareness. For that, the service customer needs to be given guarantees by the service provider on the non functional QoS attributes with which a service will be provided. Hence, service level agreement (SLA) is intended to specify (in terms of clauses) the level of such non-functional QoS delivered to the customer. In addition, the customer must be provided with tools to show the fulfillment of quality guarantees through the monitoring process. Traditionally, control over private data has been performed mainly with authorization decisions on a subject's access to target resources. However, the challenge of private data management goes beyond, i.e., knowing that the private data is already used. Thus, while the *access control* aspect of security and privacy is well understood, it is unclear how to manage *usage control*, i.e., the private data is in the hand of the provider because the provider already has an access to it (by an access control mechanism), but the provider can use it even in sensitive contexts. For that, it is necessary to control its usage. In response to the privacy concerns quoted above, in [11,32] we have proposed a privacy aware SLA that spells out a set of requirements related to customer's privacy rights in terms of how a service provider must handle privacy information. The properties and private requirements can be checked at design time prior to execution. However, the requirements monitoring has strong motivations since those properties can be violated at runtime. Therefore, checking the compliance of the usage defined in the SLA at runtime is a challenging issue. This issue must be properly addressed; otherwise, it could lead to SLA breaches and to lower service quality. For most services, any non fulfillment of QoS perceived at the customer's end can have severe consequences. For that, it is necessary to observe the Private Data Usage Flow (PDUF) when monitoring SLA behavior. Such monitoring requires to be carried out with the help of legal entities to ensure the trusted both by the service provider and customer. We define the usage compliance issue with regards to requirements, as a query containment problem, i.e., checking whether for all databases, the answer to a query is a subset of the answer of another query [12]. Querying is a fundamental mechanism for extracting data from a database, and the query containment problem has been the subject of an extensive investigation in the past years in database community [7,12–14]. In order to check the usage compliance of private data, we show that the extracted information defined as views from the usage flow are included in the information extracted from the requirements in SLA.

A common approach developed to support requirements monitoring at runtime assumes that the system must identify the set of requirements to be monitored. In fact, as part of the privacy SLA model, the set of privacy requirements to be monitored is needed from which *monitoring private units* are extracted and their occurrences at runtime would imply the violation of the requirements. Besides the functional properties (e.g. operations of the service), the time-related aspects are relevant in SLA setting. In addition, the non-compliance or failure to uphold the privacy requirements which are manifested in terms of vulnerabilities must be identified.

1.2 Overview of the monitoring framework

To illustrate our motivation, we use the on line purchase service as a use case for this paper. The level of privacy guarantee that a service provider is expected to give to a customer is specified in SLA clauses, signed by both parties. The privacy aware SLA states the rights and obligations (rights and obligations on data, Sect. 2) in terms of operations that the provider can perform on the collected private data. The latter agrees to honor those clauses during the contract validity.
In what follows, we provide some examples of the above mentioned privacy-aware SLA clauses:

- C_1: the provider must collect the private data email, contact address and credit card number (ccn) of a customer.
- C_2: retention period of customer email is equal to *15 days*.
- C_3: ccn is held until the end of each payment process.
- C_4: the provider has the right to send available product information and invitations by using email.
- C_5: the provider has the right to send the invoices by using the credit card number.
- C_6: the provider has the right to share customer address with a given company C.
- C_7: the provider has the obligation to encrypt a credit card number at the end of each payment process.
- C_8: the provider has the obligation to notify the customer that the ccn has been encrypted.

The clauses quoted above identify some of the provider behavior for managing customer private data; namely, what private data is authorized to be collected by the service, when it is used, for what purpose, and by whom. A primary objective of this paper is to show one way of monitoring the clauses' compliance at runtime, i.e., to keep track of all private data usage flow, with or without violations. Also, at any time, a trusted third party (e.g., legal entity), which is recognized by both the service provider and the customer, can be in charge of making an audit on the privacy usage flow, i.e., to verify what a service provider does with the private data and to ensure that it honors its obligations.

1.2.1 Framework

The framework for monitoring the compliance of SLA clauses is depicted in Fig. 1. The interaction between the Web service provider and the service customer is regulated by a signed contract. As soon as the provider performs an operation on private data (collected from customer), the corresponding clauses are triggered. Such activations are considered as the events recorded in the logs. To monitor the compliance of the SLA clauses, the PDUF model helps to observe the behavior of the private data usage flow: which clause is triggered or violated and when, what types of vulnerabilities happen, or which clause is compliant and so on. A PDUF model is provided to represent such a behavior (Sect. 3). A *legal entity* can be involved to make an audit on demand to guarantee privacy SLA compliance. If a violation occurs, the legal entity notifies the signatory contracting parties of violations. The issue about when and how notifications

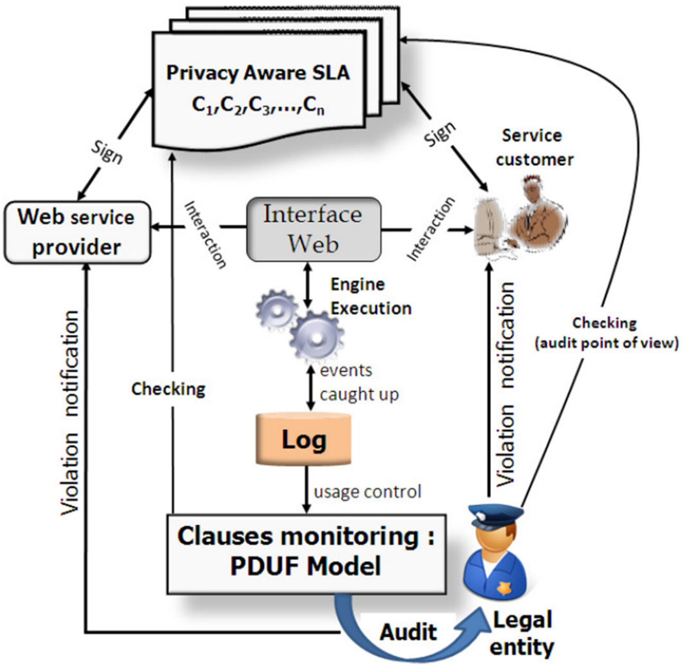


Fig. 1 The monitoring framework

of privacy violations are processed by the legal entity falls out of the scope of this work. As an example, the legal entity in France can be the *CNIL* (French Data Protection Authority)[1,2] which is an independent French administrative regulatory body whose mission is to ensure that data privacy law is applied to the collection, storage, and use of personal data. CNIL is an external auditor and is often asked for an audit.

1.3 Contributions

In this paper, we make the following contributions:

1. We propose a state machine based model to monitor the private data usage. It is called PDUF. The proposed monitoring model supports abstractions allowing the behavior observations to be expressed. It *observes* which clause is activated or violated and when, what types of violations happen, or which clause is compliant.
2. We propose a query containment reasoning mechanism to check the privacy SLA compliance. We show that the usage compliance problem can be defined as a query containment problem in databases, i.e., the answer of a query over a view on PDUF is always a subset of the answer of a query over the SLA requirements.

[1] http://www.cnil.fr/english/

[2] https://privacyassociation.org/news/a/2008-07-global-privacy-dispatches-france-cnil-annual-report/

3. We provide an implementation of our Privacy aware SLA Monitoring system called (*PaM system*). We provide an easy-to-use, and efficient surveillance tool for the purpose of dynamically observing the private data usage flow. The proposed system includes a visual language to visualize the usage flow abstracted from the PDUF.
4. We resort to a case study to evaluate the proposed system in terms of violation detections. We show that our model can effectively determine the violation periods and the compliance through the usage frequency regarding the data type.

1.4 Paper layout

The rest of the paper is organized as follows. Section 2 provides an overview of our privacy model. Section 3 presents the required ingredients for monitoring private data usage. Section 4 describes the private data usage flow model. Section 5 demonstrates that usage compliance is a type of query containment problem. Section 6 describes the architecture and implementation of the framework along with a set of experiments. We review related work in Sect. 7 and conclude with a summary and directions for future work.

2 Background: privacy model

In this section, we briefly recall our privacy model for Web services presented in [11,32]. The privacy model identifies relevant abstractions defined in terms of the following requirements:

- *Data-right*, is a predefined operation on data that the service is authorized to accomplish. We distinguish two types of operations (i) operation used to complete the service activity for the current purpose for which it was provided and denoted by $op_{current}$ (ii) operation used by a service to achieve other activities than those for which it was provided, called $op_{extra\text{-}activity}$.
- *Data-obligation*, is the expected action to be performed by the service provider (respectively the third parties) after handling private data in data-right. This type of obligation is related to the management of personal data in terms of their selection, deletion or transformation.

Formally speaking, we define data-right and data-obligation as follows:

Definition 1 (*Data-right*) A data-right r_d is a tuple (u, d, op, μ_{rd}), where: (*i*) $\mathcal{U}$ is the ontology of authorized users (service provider, third parties) and $u \subseteq \mathcal{U}$, (*ii*) $\mathcal{D}$ is the ontology of private data and $d \subseteq \mathcal{D}$, (*iii*) $\mathcal{OP}$ is the set of authorized operations identifying the purposes of the service and $op \subseteq \mathcal{OP}$, (*iv*) μ_{rd} is the period of data retention (the data-right validity), and $\mathcal{R}^d$ is the set of data-rights with $\mathcal{R}^d = \{\{r_d^i\} / i > 0,\ i \in N\}$.

Example 1 1. $r^1_{email}(sp, email, send\ Invoice, \mu_{r1email})$, specifies that the provider *sp* has the right to use an *email* for sending invoices during the period $\mu_{r1email}$.

2. $r^2_{email}(sp, email, send\ invitation, [d_s, d_s + 1month])$, specifies that the provider *sp* has also the right to use an *email* for the marketing purpose, namely sending the invitation, during the period $\mu_{r2email}$ which is $1month$ after both sides have signed the contract at d_s date.
3. $r_{ccn}(sp, ccn, payment\ Invoice, \mu_{rccn})$, specifies that the provider *sp* has the right to use *ccn* for the invoice payment during the period μ_{rccn}.
4. $r_{add}(sp, address, delivering, \mu_{radd})$, specifies that the provider *sp* has the right to use *address* for delivering the products during the period μ_{radd}.

Definition 2 (*Data-obligation*) A data-obligation o_d is a tuple (u, d, a_o, μ_{od}), where: (*i*) $\mathcal{U}$ is the ontology of authorized users and $u \subseteq \mathcal{U}$, (*ii*) $\mathcal{D}$ is the ontology of private data and $d \subseteq \mathcal{D}$, (*iii*) $\mathcal{A}_o$ a set of security actions that must be taken by the user and $a_o \in \mathcal{A}_o$, (*iv*) μ_{od} is an activation date of the obligation, and $\mathcal{O}^d$ is the set of data-obligations with $\mathcal{O}^d = \{\{o^i_d\} \ / \ i > 0 \ , \ i \in N\}$.

Example 2 1. $o_{add}(sp, address, delete, \mu_{oadd})$, specifies that the provider *sp* must delete the address for a given data subject at μ_{oadd}, i.e., when the authorization of address retention time is elapsed.
2. $o^1_{ccn}(sp, ccn, encrypt, [d_{pay} + 1day])$, specifies that the provider *sp* must encrypt the *ccn* for a given data subject at the end of each payment process, for instance, at d_{pay}+1 day (μ_{occn}).
3. $o^2_{ccn}(sp, ccn, notify, \mu_{occn})$, specifies that the provider *sp* must notify the customer that the ccn has been encrypted at a time point t.

On the basis of these requirements, we formalize a privacy data model as follows:

Definition 3 (*A privacy data model*) A privacy data model $\mathcal{D}^p$ is a tuple $< \mathcal{R}^d, \mathcal{O}^d >$, where $\mathcal{R}^d$ is the set of data-rights and $\mathcal{O}^d$ is the set of data-obligations.

Using the proposed privacy model, we extend the WS-Agreement specification which does not support the privacy structure, and does not include the possibility of updating a contract at runtime. A guarantee is not fulfilled because of an event occurring in the service behavior, and may change the personal data use. The proposed extension is reflected in a new component in the WS-Agreement called *Privacy-aware SLA*. It is structured into two levels:

1. *The policy level* specifies the *Privacy-Data term* defined as a set of *clauses* $\mathcal{C}$ stipulated in the contract agreed on by the provider and the customer. $\mathcal{C} \subseteq \mathcal{R}^d \bigcup \mathcal{O}^d, d \in \mathcal{D}$.
2. *The negotiation level* specifies all the possible events that may happen in the service behavior, and thus evolves the privacy guarantee terms that are defined in the policy level. Negotiation terms are all possible actions to be taken if the guarantee of privacy terms is not respected, and a conflict arises. These are used through a negotiation protocol between the service provider and the customer.

We also define in this level the validity period of the contract and a set of penalties, if the requirements are not fulfilled. More details can be found in [32].

In this paper, our focus is on the first level. We present a way of observing the usage of the private data at runtime, and show how to capture the usage compliance regarding the clauses.

3 Private data usage flow model

In this section we model the activation of SLA clauses on the basis of observations of the PDUF. Such behavior observations are expressed by means of previous abstractions (data-right, data-obligation) and artifacts allowing the runtime monitoring defined below.

3.1 Privacy requirements for monitoring

One of the key aspects for service reliability is the trustworthiness of the compliance of its collected private data usage with regard to the contract. To ensure the usage compliance, observations of the service behavior and its private data usage become a necessity. To make such an observation effective, two essential ingredients are required. First, we need to identify what kind of knowledge must be monitored, and then the knowledge which makes the contract non-compliant. In this section we discuss both aspects.

3.1.1 Monitoring private units

We distinguish four types of private units to be monitored: *private data unit*, *operation unit*, *role unit* and *temporal unit*.

- *Private data unit* The private data unit d is the core of our monitoring framework. It includes any data item deemed confidential by the customer. In fact, from the log we need to observe only the private data and its usage.
- *Operation unit* We distinguish three types of operations (i) operation used to complete the service activity for its current purpose for which it was provided and denoted by $Op_{current}$ (ii) operation used by a service to achieve other activities than those for which it was provided, called $Op_{extra\text{-}activity}$ (iii) actions performed by the provider after handling personal information in data-right, called *security-action*. The first two kinds of operations are proposed in order to know when a usage compliance is compromised, while the service runs. The set of authorized operations is denoted by $\mathcal{OP}$.
- *Role unit* We need to observe who will use the private data. We denote the set of authorized users by $\mathcal{U}$.
- *Temporal unit* The analysis of time-related aspects of the privacy monitoring requires the specification of operation durations and timed requirements. We identify four temporal units (discussed in the following), and we denote the set of temporal units by $\mathcal{T}$:

Definition 4 (*Right triggering time*) For each collected private data d, the right triggering time denoted ϵ_{rd} is the activation time of the operation associated with the right:
$\forall r_d^i \in \mathcal{C} \rightarrow \exists\, \epsilon_{rd}^i \in T \mid (r_d^i.op)^{\epsilon_{rd}^i}$ is activated, where i is the ith right associated with the private data d, $(r_d^i.op)$ means the operation included in the ith right related to d, $\mathcal{C}$ is a set of clauses in the SLA, and T is a domain of time values.

Definition 5 (*Right end time*) For each collected private data d, the right end time denoted β_{rd} is the end time of the data use (operation) associated with the right: $\forall r_d^i \in \mathcal{C} \rightarrow \exists\ \beta_{rd}^i \in T \mid (r_d^i.op)^{\beta_{rd}^i}$ is finished.

Definition 6 (*Obligation triggering time*) For each collected private data d, the obligation triggering time denoted μ_{od} is the activation time of the security action associated with the obligation:
$\forall o_d^j \in \mathcal{C} \rightarrow \exists\ \mu_{od}^j \in T \mid (o_d^j.a)^{\mu_{od}^j}$ is activated when $(max\ (\beta_{rd}^i) \leq \mu_{od}^j)$ is true, where β_{rd}^i is the ith right end time associated with the right r_d^i (the security action is triggered when each right associated with the obligation is achieved).

Definition 7 (*Obligation end time*) For each collected private data d, the obligation end time denoted α_d is the end time of the action associated with the obligation: $\forall o_d^j \in \mathcal{C} \rightarrow \exists\ \alpha_d^j \in T \mid (o_d^j.a)^{\alpha_d^j}$ is ended.

3.1.2 Privacy violations

In this section, we identify the non-compliance or failure to uphold the contract, manifested in terms of violations. We define a list of privacy violations which are most likely throughout the private data usage. We have classified these vulnerabilities into two classes, *explicit* and *implicit violations*. The former can be visualized in our private data usage flow model (Sect. 3.2), whereas the latter cannot be identified clearly. For instance, *security on data* (e.g., delete, update, hide, unhide,...) and *accountability* cannot be identified in our model; so, implicit violations are not within the scope of this paper. We classify three types of explicit violations, *usage violation*, *role violation* and *temporal violations*:

– *Usage violation* A usage violation is an unauthorized operation on the collected private data *d*. The violation occurs when the activated operation $(r_d^i.op_d^i) \notin \mathcal{OP}$. i is the ith unauthorized operation associated with the private data *d*. The usage violation is denoted $op_{d\ wrong\text{-}use}^i$ in PDUF model (Sect. 3.2).
– *Role violation* A role violation is an unauthorized service user or third party which does not have the right to use the private data *d*. It happens when $R_d^i.u_d^i \notin \mathcal{U}$. i is the ith unauthorized service user of the private data *d*. The role violation is denoted by $u_{d\ wrong\text{-}user}^i$.
– *Temporal violation* We identify two temporal violations, *retention violation* and *obligation violation*:
  – *Retention violation* A retention violation is a vulnerability on the right data retention period. It happens when $(\beta_{rd}^i - \epsilon_{rd}^i > \mu_{rd}^i)$ is true, where β_{rd}^i is the ith right end time and ϵ_{rd}^i is the ith right triggering time associated with the right r_d^i and μ_{rd}^i is the ith authorized period of data-right r_d^i.
  – *Obligation violation* An obligation violation is a violation on the obligation triggering time. It happens when $(max\ (\beta_{rd}^i) > \mu_{od}^j)$ is true, where β_{rd}^i is the ith right end time associated with the right r_d^i (so, we take the ith longest end time) and μ_{od}^j is the jth authorized activation date of data-obligation o_d^j associated with the private data *d*.

Note that the defined violations are not complete, and at runtime some new violations may be detected, and thus added to the violation database. In fact, these new violations can be detected (e.g., using reasoning mechanisms) as an SLA is dynamic. However, the service SLA evolution is out of the scope of the paper.

3.2 Private data usage flow

The PDUF is expressed by using a state machine thanks to its formal semantics, which is well suited to describe the activation of different clauses of the contract. It will show *which* clause is activated or violated, and *when*, and also specify the states of each activation clause in the SLA. The time-related requirement properties set in the contract are depicted explicitly in the state machine. The semantics of the state machine is to define all the triggered operations involving the private data from the SLA activation (initial state) to the end of the SLA (final state). To capture this semantics, the possible usage of the private data is represented through a set of paths. Figure 2 depicts a sample of the PDUF for the purchase service provider, explained further in Example 3 in the following. Its shows only three *usage flow paths* of the email data, (1) the first path $A - B - C_1 - D_1 - E_1 - F_1$ (Fig. 2(1, 2)) the second $A - B - C_2 - D_2 - E_2 - F_2$ (Fig. 2(2, 3)) and the third path flow $A - B - C_3 - D_3$ (Fig. 2(3)), the first one is described below.
We have identified several abstractions in relation to private data flow, *private data usage* abstractions and *authorization* abstractions. The first abstractions describe the different states in which the contract is: which data is collected, when it is collected, for what it is used, and who uses it. The authorization abstractions provide the conditions that must be met for transitions to be triggered.
In this formalism, the fact that the private data has a time retention for a right (respectively the activation time of an obligation) called *fixed guard time*, the data usage time is represented by a time increment in the state, followed by the end of the right (respectively the obligation) with success or violation of that time. Intuitively, PDUF is a finite state machine for which a set of clock variables Δ is assigned. A clock variable is assigned for each activation of the clauses (rights and obligations). The values of these variables increase through time. The transition takes place when an operation is triggered or time units are activated. If the temporal units are compliant to the guard times, it will invite the transition to take place with success and no violation is recorded in that state. However, if non-compliance is detected, the transition will take place with violation, and the state is marked as violated.

Definition 8 (*PDUF*) A PDUF is a tuple

$$(\mathcal{S}, s_i, s_f, s_{fail}, \mathcal{M}, \mathcal{R}, \mathcal{Q})$$

- $\mathcal{S}$ is a set of states;
- $s_i \in \mathcal{S}$ is the initial state,$s_f \in \mathcal{S}$ is the final state and $s_{fail} \in \mathcal{S}$ is the failed state;
- $\mathcal{M}$ is a set of monitoring private units: set of triggered operations and/or set of temporal units, $\mathcal{M} = \{OP, \mathcal{T}\}$;

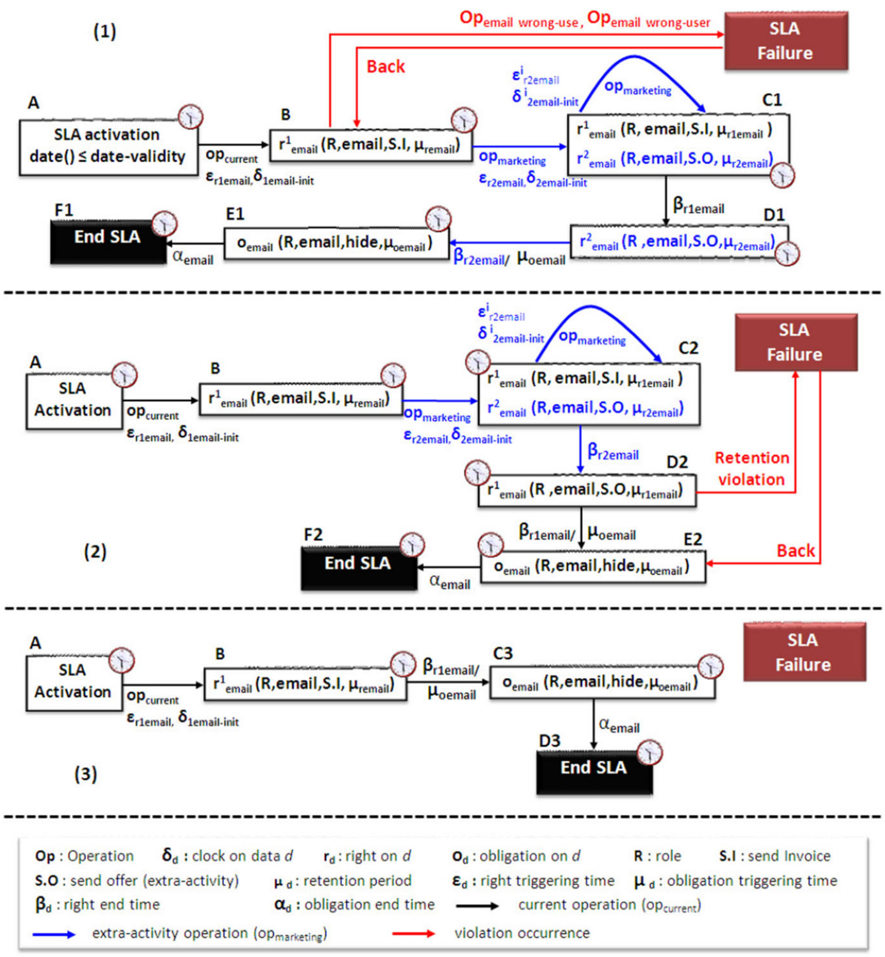


Fig. 2 PDUF: email usage flow paths

- $\mathcal{R} \subseteq \mathcal{S}^2 \times \mathcal{M} \times 2^{\Delta}$ is a set of transitions with a set of operations or a set of triggering times and a set of clocks to be initialized $\delta_{d-init} \in \Delta$ (set of clock variables);
- $\mathcal{Q} : \mathcal{S} \rightarrow \{\delta_i \mid \delta_i \in \Delta, i \geq 1\}$ assigns a set of clocks to the states.

The effect of each transition $\mathcal{R}(s, s^{'}, m, c)$ from the source state s to the target state $s^{'}$ is to set a status of the clause in the SLA which means to perform an operation $op \in \mathcal{OP}$ using a private data or a monitoring time unit $t \in \mathcal{T}$ is activated.

Definition 9 (*Covered SLA*) Let $\mathcal{PF} = s_i, s_1, s_2, \ldots, s_j$ be a path flow from the initial state s_i to s_j, where $s_i, s_j \in \mathcal{S}$ *and* $s_k \in \mathcal{S}, \forall k \geq 1$. $\mathcal{PF}$ is called a partially covered SLA iff $\forall s_k \in \mathcal{PF}$ and $s_k \neq s_{fail}$, and $\mathcal{PF}$ is called a covered SLA if $s_j = s_f$.

The covered SLA is suggested in order to measure the QoS of the services. If the SLA is partially covered, it means there are some violations. In this regard, we use

our previous works [37,54] where we introduced the notion of penalty. So, if a service does not match totally the clauses stipulated in the SLA, some penalties can be applied against it. The penalty aspect is however out of the scope of this paper.
Let us define PDUF semantics through the following example.

Example 3 (PDUF sample for the purchase service) Let us consider our example of a purchase service (Sect. 1.2) (Fig. 21). After a privacy aware SLA has been signed between the two parties, a set of clauses with a validity period denoted by *validity-date* is set. The clauses are specified as follows: at the date *date()* the contract is activated and the service provider collects the emails. The private data is used for two types of operations **(1)** to complete the service activity for the current purpose i.e. the email is used to send invoices. The operation is expressed by the following right $r^1_{email}(R, email, S.I, \mu_{r1email})$, **(2)** to achieve other activities than those for which it is provided, such as marketing purpose i.e. the email is used to send information regarding the available products and their prices; the clause is expressed by the right $r^2_{email}(R, email, S.O, \mu_{r2email})$. When the retention times of the private data email ($\beta_{r1email}$, $\beta_{r2email}$) are elapsed, the corresponding obligation is triggered, $O_{email}(R, email, hide, \mu_{oemail})$. The obligation specifies that the role R must hide the private data (email) as soon as the activation date μ_{oemail} is reached.

States: We define four types of states:

1. The initial state s_i represents the activation of the SLA where the first private data item (email) of the customer is collected. In Fig. 2(1), s_i is defined by A.
2. The intermediary states represent the flow of the collected private data use. The activation of a new state is caused by:
   - The use of data to complete the activity of the service for which it was provided, identified in Fig. 2(1) by $Op_{current}$ (black line). In state B, the current operation is *Send Invoice*. In this state, the clock δ_{1email} is activated to r^1_{email} and incremented through a time.
   - The use of data to achieve an extra-activity depicted in Fig. 2(1) by $Op_{marketing}$ (blue line). The right r^2_{email} is activated in the state C_1 as soon as the marketing operation is triggered. The same operation can be activated as many times as the time retention $\mu_{r2email}$ is valid. It is represented by a *loop* in state C_1. The contract remains in the same state.
   - The data usage is finished (the right). For instance, the contract will be in state D_1 since data retention guard time is reached, which means the finishing time of the right denoted by $\beta_{r1email}$ is over.
   - The activation of an operation dealing with the security (obligations) when the retention time of the private data defined as a fixed time in the right is elapsed and the time for triggering the obligation starts. For instance, such case is depicted in state E_1, where o_{email} is triggered when the usage times $\beta_{r1email}$ and $\beta_{r2email}$ are reached and the obligation time starts, defined in the transition by μ_{oemail}.
3. The *virtual* state labeled *SLA Failure* will be reached when private data is used to achieve the usage violation, and/or role violation and/or temporal violations happen regarding the clock variable values and fixed times. For instance, the first type of violation is identified by $Op_{email\ wrong\text{-}use}$ (usage violation) (red line) or

$op_{email\ wrong\text{-}user}$ (role violation) between state *B* and *SLA Failure* state. We call this state as a virtual state because it is considered only like a flag of the violations.
4. The final state s_f represents the end of the SLA which means the validity of the contract is over, and either the data usage in all its shapes are compliant to the SLA or they are not respected due to the violations. The best case is to reach the end of the SLA without any violation, as depicted in Fig. 2(3) from state *A* to the 'End SLA' state.

Transitions: Transitions are labeled with conditions which must be met for the transition to be triggered. We identify three kinds of authorization abstraction conditions:

– Activation conditions. Regarding the operation unit, we define two types of activation conditions (1) $op_{current}$ condition, for instance from state *A* to state *B*, authorizes an operation (e.g., send invoice) to collect private data (e.g., email) to achieve the current aim of the service, (2) $op_{marketing}$ condition from state *B* to state C_1, authorizes the operation dealing with an extra-activity of the service to be triggered (e.g., send offer).
– Temporal conditions. The transition is called *timed transition*. Regarding the temporal units, we define four types of timed transitions (1) *right triggering time* ϵ_{rd}, for instance from state *B* to state C_1 the timed transition is labeled by $\epsilon_{r2email}$ along with the activation of the clock δ_{2email} assigned to the right r^2_{email} (2) *Right end time* β_{rd}, from state C_1 to state D_1 the transition is labeled $\beta_{r1email}$, which means the send invoice operation is over (3) *Obligation triggering time* μ_{od}, the authorization to keep the private data is finished and the obligation is triggered, for instance from state D_1 to E_1, the transition is labeled μ_{oemail}, the security action must be fired (4) *Obligation end time* α_d, the obligation is over, for instance from state E_1 to the End SLA state, the transition is labeled α_{email}.
– Violation Conditions. The transition can be labeled by all the violations identified in Sect. 3.1.2. For the violation dealing with the operations, the target state of the transition is *SLA Failure* and *Back* to the previous state, such as, the operation $op_{email\ wrong\text{-}use}$ on the transition between state *B* and the *SLA Failure* state, and back to state *B*. For the temporal violations the target state of the transition is *SLA Failure* and no back tracking is allowed, but to the next state. For instance, a time violation happens in D_2 (retention violation) and the system passes to the next state E_2 (Fig. 2(2)).

In the following, we give the details of the path flow depicted in Fig. 2(1).

Example 4 (email usage flow paths) The provider is authorized to collect email data to achieve the current aim of the service (the right r^1_{email}). The transition from state *A* to state *B* is labeled $op_{current}$ with the timed transition denoted by $\epsilon_{r1email}$ (right triggering time) along with the activation of the corresponding clock δ_{1email}. Similarly, the marketing operation $op_{marketing}$ is triggered from state B to state C_1. In state C_1, two clauses of the SLA are triggered (1) the *current* operation r^1_{email} (send invoice) is still activated and is accumulated from the previous state *B* because the retention time of the right r^1_{email} associated with the data *email* is not elapsed (2) the *send-offer* operation (r^2_{email}) is activated by entering C_1 for marketing purpose of the service (not to complete the service); it is an extra-activity of the service.

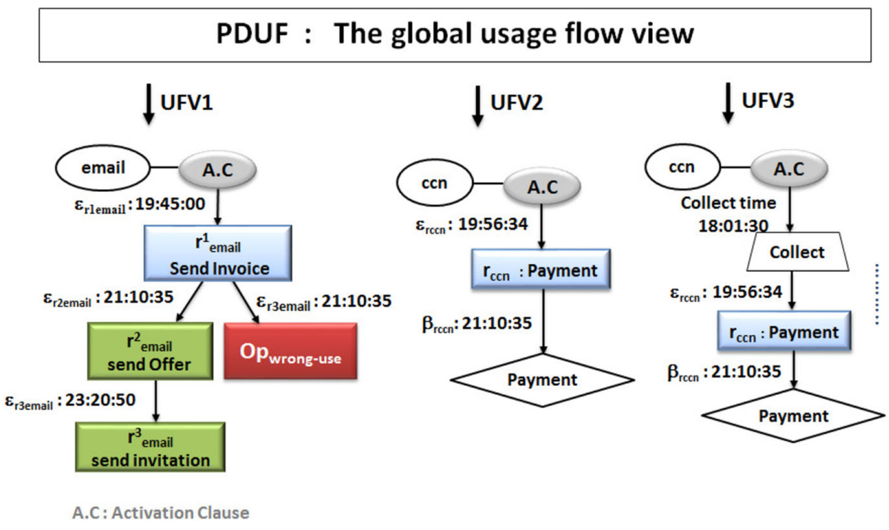


Fig. 3 Usage flow views

In state D_1 one clause is still activated, the extra-activity r^2_{email}, and then accumulated in the new state from C_1. From state C_1 to state D_1 the transition is labeled $\beta_{r1email}$, which means the send invoice operation associated with the right r^1_{email} is over. Similarly, from state D_1 to state E_1 the transition is labeled $\beta_{r2email}$, which means the send offer operation associated with the right r^2_{email} is over, in state E_1 the obligation o_{email} is activated because the time μ_{oemail} to activate is reached.

4 Usage views

As mentioned earlier, one of our aims is to monitor the private data usage flow by observing the behavior of the service (PDUF). The idea is to use the concept of *view* [14] to capture a part of the activated clauses in the *PDUF*. Next, we show how to capture the compliance of the usage views extracted from the PDUF.

4.1 Usage flow view

For checking the *usage compliance*, we introduce the notion of *Usage Flow View* (UFV), a part of activated clauses in the *PDUF* global process. *PDUF* is considered as a log storing all the usage performed on personal data. The UFV is a query on the log for specific private data. In fact, PDUF works as a global knowledge base, denoted by *PDB*, from which we extract the different views (Fig. 3), where,

– The view *UFV1* provides all the operations executed on *email* data within the corresponding triggering time,
– *UFV2* displays the execution time of each operation executed on *ccn* data,
– *UFV3* shows the *ccn* retention time. It is a time from which the data is collected up to the end of the last 'right execution'.

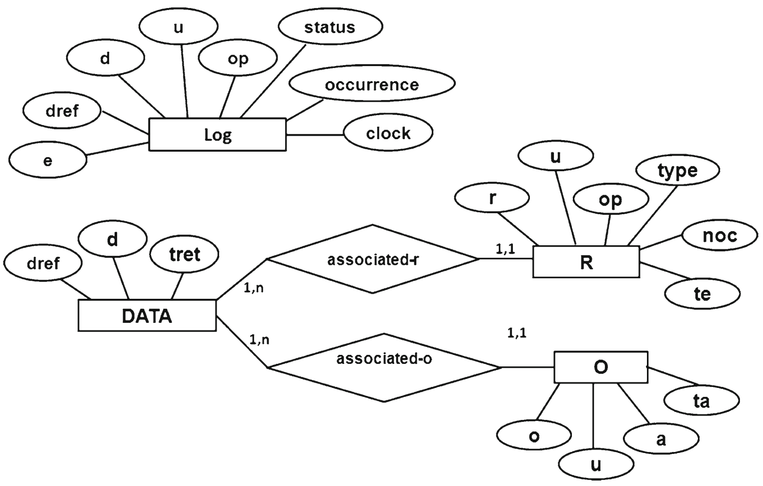


Fig. 4 Graphical representation of PDB database

The symbols in Fig. 3 are related to different actions to be considered in the PDUF. For a better visualization of the PDUF, we introduce a visual language, with further details provided in Sect. 6.2.
The database *PDB* (Fig. 4) contains four relations; all the attributes of the relations should be defined as an element in the monitoring private units $\mathcal{M}$ set. The relations are:

1. The Log relation (PDUF) stores all the executed operations as events. We need to distinguish between two kinds of events: (1) the collect event denoted by *Col-E* which is designed to collect the private data and has an activation time, *the collect-time*, (2) The events on data (data-right, data-obligation). Each event can have one of two statuses, *Act-OP* for the triggering event and *End-OP* for the completion of the event. The attributes of the Log relation are:
   - e is a unique identifier of the event.
   - d_{ref} is the identifier of the used private data.
   - d is the value of private data.
   - u is the identifier of service user.
   - *op* is the event name (the right/obligation or data collect event).
   - *status* represents the type of the event (Col-E, Act-OP, End-OP).
   - *occurrence* is the execution number of the right (if the right is triggered many times).
   - *clock* is the triggering time (respectively, data collect time) or the end time of the event.
2. The requirement relation $DATA(d_{ref}, d, t_{ret})$ stores the authorized retention time t_{ret} of d. The identifier of the relation is d_{ref}.
3. The requirement relation $R(r, d_{ref}, u, op, type, n_{oc}, t_e)$ stores the right private units specified in the contract with their corresponding type (current,extra-activity,

etc.) and the authorized number of their occurrences n_{oc}, t_e is the authorized execution time of an operation *op*. The identifier is the right *r*.

4. The requirement relation $O(o, d_{ref}, u, a, t_a)$ stores the obligation private units. t_a is the authorized activation time of a security action *a* while the identifier is the obligation *o*.

The UFV is used to query the Log relation for providing various information on data *d*. The information items are: service user, executed right/obligation, event triggering times, event end times and data collected time. For instance, the following UFV asks for all the usage executed on email:

```
CREATE VIEW UFV as
SELECT  e, d_ref, d, u, op, status, occurrence, clock
FROM  Log  WHERE  d='email';
```

However, the results provided by UFVs are considered incomplete. In the next section, we provide more details on this, and specifically, on information regarding the handling of temporal aspects.

4.2 Usage temporal views

The information provided by the UFVs is considered incomplete for checking the usage compliance adequately. Indeed, the process needs more temporal information which can be gleaned from the UFV results. The additional information consists of the execution time, the number of occurrences of each right (operation), and the retention time of data *d*. We define Algorithm 1 to compute this additional temporal information, which is to be applied for each data *d*. The algorithm generates two views including the usage and temporal aspects:

- The *Usage Duration View*, $UDV(d_{ref}, op, u, t_e, n_{oc})$ stores the execution time t_e and the number of occurrences n_{oc} of each executed right *op* (obligation). It is important to know that if the right is activated many times, we keep the maximum of all execution times of the right occurrences.
- The *Retention Data View*, $RDV(d_{ref}, d, t_{ret})$, stores the retention time of data, since its collection.

On the basis of the usage views, we define the usage compliance approach in the following.

5 Usage compliance based query containment

Let us recall a useful definition (query containment) to understand the rest of the section. Query containment is the problem of checking whether for every database, the result of one query is a subset of the result of another query [1,7,12–14,19,20,22, 23,29,40]. In the following definition, we denote the result of computing the query Q over the database D by Q(D).

Algorithm 1 Usage Temporal Views Algorithm

```
Require: UFV related to the data d ($UFV_d$)
Ensure: Usage Temporal Views (UDV, RDV)
1: get t.collect time ($t \in UFV_d$) {the first tuple gives the collect time of $d$}
2: for each t.op ($t \in UFV_d$) do
3:     find t.op.occurrences, t.op.triggering times and t.op.end times
4:     calculate max(op.occurrences)
5:     calculate op.execution times        {end times - triggering times}
6:     calculate max(op.execution times)
7:     store in UDV
8: end for
9: calculate retention time            {last end time - collect time}
10: store in RDV
```

Definition 10 (*Query containment and equivalence*) A query *Q1* is said to be contained in a query *Q2*, denoted by $Q_1 \subseteq Q_2$, if for all databases D, the set of tuples computed for *Q1* is a subset of those computed for *Q2*, i.e., $Q_1(D) \subseteq Q_2(D)$. The two queries are said to be equivalent if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$.

We now consider a database *PDB* and two queries Q_r and *UFV*, where Q_r is over the requirements table R and *UFV* is over the log table. Informally speaking, the problem of the *usage compliance* is similar to the *query containment* problem: the query *UFV* is contained in the query Q_r, denoted by $UFV \subseteq Q_r$, if the answer to *UFV* is a subset of the answer to Q_r for *any* database instance. Then, the *usage compliance* means the *log compliance*.
Now, if we consider the temporal information extracted from the UFV, we can say that the *usage compliance* is the problem concerned with checking whether the results that the queries obtain from the usage temporal views (UDV, RDV) are subset of the answers obtained by another query on the same database (requirement tables). In other words, finding the exact-tuples-match, or more precisely, finding the exact-monitoring units-match based queries results set. Now, the database is accessible only through the views. Consider the queries UVF on *Log*, Q_{udv} on *UDV*, Q_{rdv} on *RDV*, Q_r on requirement table R and Q_d on requirement table DATA (Fig. 5). Formally, the *usage compliance* holds if two conditions are satisfied:

1. *The operation compliance* requires that a tuple belonging to the result set of Q_{udv} should also belong to the result set of Q_r. This means that any information in Q_{udv} should appear in Q_r.
2. *The retention compliance* requires that a tuple belonging to the result set of Q_{rdv} should also belong to the result set of Q_d. This means that any information in Q_{rdv} should appear in Q_d.

Definition 11 (*Operation compliance* $\subseteq_{PC}$) Let Q_{udv} be the result set on UDV and Q_r be the result set on requirement table R. Q_{udv} is operation compliance contained in Q_r, written $Q_{udv} \subseteq_{PC} Q_r$, if and only if $\forall t \in Q_{udv}, \exists t' \in Q_r / Q_{udv}(t) = Q_r(t')$.

Definition 12 (*Retention compliance* $\subseteq_{RC}$) Let Q_{rdv} be the result set on RDV and Q_d be the result set on requirement table DATA. Q_{rdv} is retention compliance contained in Q_d, written $Q_{rdv} \subseteq_{RC} Q_d$, if and only if $\forall t \in Q_{rdv}, t' \in Q_d \ / \ Q_{rdv}(t) = Q_d(t')$.

Fig. 5 Queries on requirement tables and usage temporal views

Select d_{ref}, u, op, n_{oc}, t_e
From UDV
(Q_{udv})

Select d_{ref}, t_{ret}
From RDV
(Q_{rdv})

Select d_{ref}, u, op, n_{oc}, t_e
From R
Where d='email'
(Q_r)

Select d_{ref}, t_{ret}
From Data
Where d='email'
(Q_d)

Theorem 1 *The usage compliance holds for data d:*

$$UFV \subseteq Q_r \quad \Leftrightarrow \quad Q_{udv} \subseteq_{PC} Q_r \quad and \quad Q_{rdv} \subseteq_{RC} Q_d.$$

The containment relationship between the queries should hold for all the private data specified in SLA. Over the years, the researchers have investigated in depth checking the query containment problem. For instance in [13] they make use of a reduction of query containment to a problem of unsatisfiability in a variant of Propositional Dynamic Logic, called cpdlg. In [25] query conjunctive problem is regarded as constraint satisfaction problem. In [19,20], they used Constructive Query Containment (CQC) method to check query containment. The aim of the CQC method is to construct a counterexample that proves that the query containment relationship being checked does not hold. The provided proof of the Theorem 1 is based on the non-containment developed in [19,20]. One suitable way of checking compliance is to check the lack of containment, *non-containment* [19,20]. Given the above queries, the way of checking compliance process is to find all the columns (monitoring units) where the assumed containment relationship between queries does not hold. Note that the unmatched columns identify the different violations on monitoring units. If the process succeeds, non-containment is proved. Thus, we will reason by contradiction in order to prove the compliance.

Proof The non-containment is proved when trying to determine whether $Q_{udv} \subseteq_{PC} Q_r$ and $Q_{rdv} \subseteq_{RC} Q_d$ are false for each private data d, that is, to find just one column (monitoring unit) where the containment relationship that we want to check does not hold. It suffices to verify that the two conditions are true:

1. The non-containment relationship between Q_{udv} and Q_r, denoted by $\neg(Q_{udv} \subseteq_{PC} Q_r)$, holds if and only if $\forall t \forall x/((x \in Q^t_{udv}(i)) \wedge (x \notin Q^t_r(i)))$ where $i \in (u, op, n_{oc}, t_e)$.
2. The non-containment relationship between Q_{rdv} and Q_d, denoted by $\neg(Q_{rdv} \subseteq_{RC} Q_d)$, holds if and only if $\exists x/((x \in Q_{rdv}(t_{ret})) \wedge (x \notin Q_d(t_{ret})))$.

□

Condition (1) gives for each column all the usage in Q_{udv} that are not specified in Q_r. Then, the violations hold and we keep the misuses in the violation table.
Condition (2) verifies if the retention time in Q_{rdv} is different from that specified in Q_d.

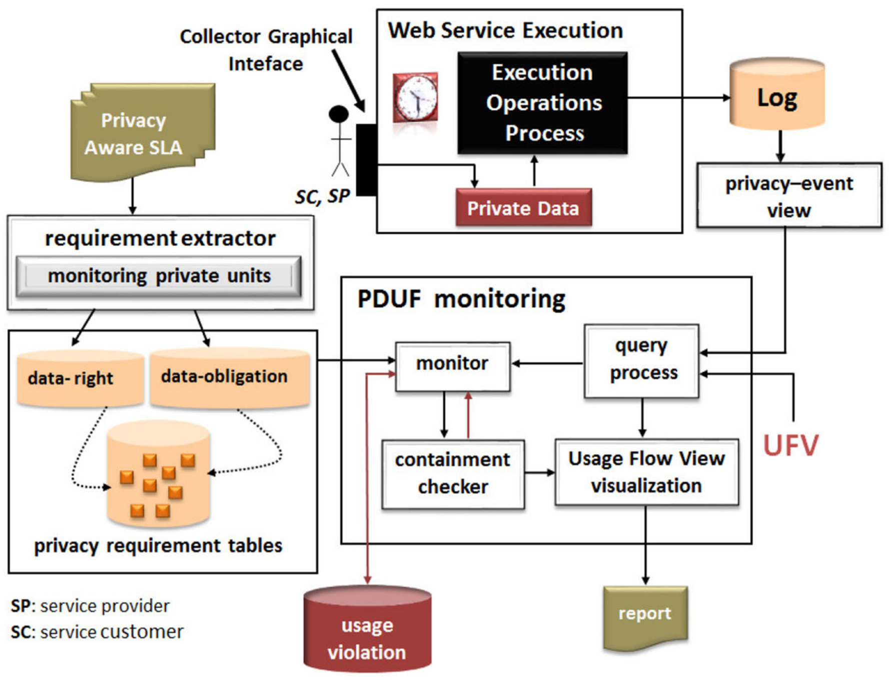


Fig. 6 The monitoring system

6 Architecture and implementation

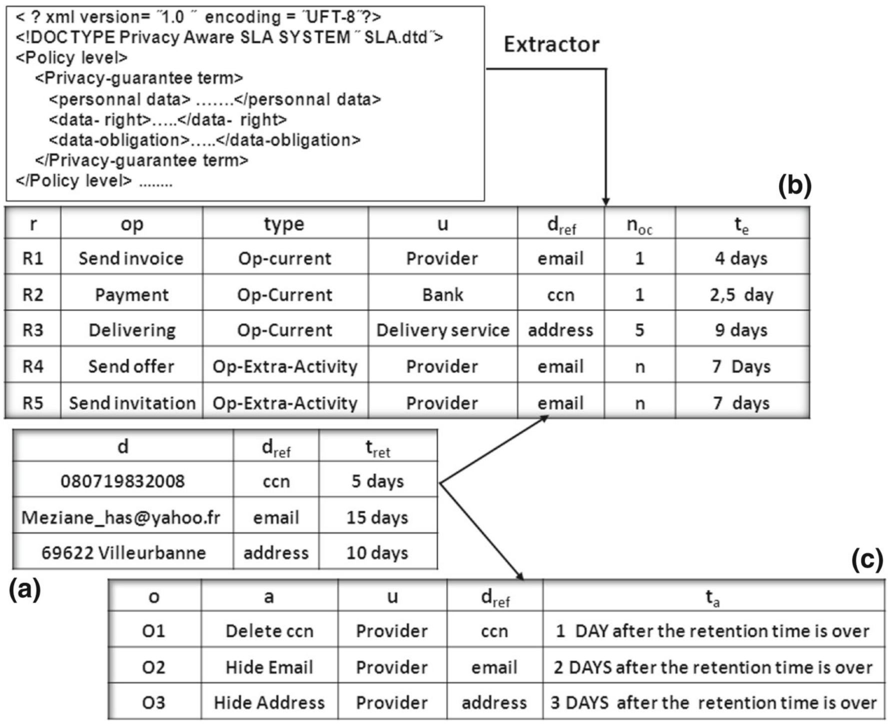
In this section, we describe the architecture and implementation details of the Privacy aware SLA Monitoring system.

6.1 Architecture

The system architecture comprises four main components (Fig. 6), namely a *requirement extractor*, *Web service execution*, *privacy-event view* and *PDUF monitoring*.

– *Requirement extractor* takes as input the contract represented in an XML-based format, and extracts the monitoring private units to be monitored from the requirements specification (rights, obligations). The private units are recorded in right and obligation tables (Fig. 7).
– *Web service execution* is an environment that manages all interactions of the Web services, which are, service client, service provider and partner services (bank service, delivery service and maintenance service). It includes two sub-components, the *execution operations process* and the *collector*.

  The *execution operations process* executes a set of operations (whether or not using personal data), to fulfill the service requirements. *Two kinds of operations are specified*: (1) *internal operations*, (2) *external operations*. The former are



```
< ? xml version= "1.0 " encoding = "UFT-8"?>
<!DOCTYPE Privacy Aware SLA SYSTEM " SLA.dtd">
<Policy level>
  <Privacy-guarantee term>
    <personnal data> .......</personnal data>
    <data- right>.....</data- right>
    <data-obligation>.....</data-obligation>
  </Privacy-guarantee term>
</Policy level> ........
```

(b)

r	op	type	u	d_{ref}	n_{oc}	t_e
R1	Send invoice	Op-current	Provider	email	1	4 days
R2	Payment	Op-Current	Bank	ccn	1	2,5 day
R3	Delivering	Op-Current	Delivery service	address	5	9 days
R4	Send offer	Op-Extra-Activity	Provider	email	n	7 Days
R5	Send invitation	Op-Extra-Activity	Provider	email	n	7 days

(a)

d	d_{ref}	t_{ret}
080719832008	ccn	5 days
Meziane_has@yahoo.fr	email	15 days
69622 Villeurbanne	address	10 days

(c)

o	a	u	d_{ref}	t_a
O1	Delete ccn	Provider	ccn	1 DAY after the retention time is over
O2	Hide Email	Provider	email	2 DAYS after the retention time is over
O3	Hide Address	Provider	address	3 DAYS after the retention time is over

Fig. 7 Requirement tables. **a** Private data table. **b** Right private units table. **c** Obligation private units table

executed by the provider while the latter are executed by the partner services. Table 1 summarizes such operations.

The *collector* is an editing interface, which allows the provider to collect the private data from the attached operation when executed.
The private data collection and the execution of the operations are considered as events provided by the Web service execution. These events are then stored in a log. It is important to know that a clock variable is assigned to each activated operation, and each collected data. The values of these variables increase through time.

– *Privacy-event view* receives the events and identifies their types and relevance to the privacy requirements being monitored, and sends them to the *query process*. All the non-relevant events are not tackled.
– *PDUF monitoring* observes the behavior of the private data usage flow by using four sub-components: a *query process*, *monitor*, *containment checker* and *usage flow view visualization*.

Query process proceeds in two steps. First, it executes the UFV, which is written via an editor, to filter only the relevant events stored in the log that meet specific criteria. Then, it generates the usage temporal views, *UDV* and *RDV* from UFV results, on the basis of the algorithm provided previously.

Table 1 Specification of operations used in the web service execution

Status operation	Type operation	Description	Example
Collect event *Col-E*	Internal	Collection of the private data	Collection of email, ccn, address
Invocation current-operation Act-$OP_{current}$	Internal	Activation of the current operation	Execution of send invoices by using email
Invocation extra-operation Act-$OP_{e}xtra\text{-}a$	Internal	Activation of the extra-activity which can be executed concurrently with the current operation	Execution of send offer or send invitation by using email
Receive operation *Rec-OP*	External	Partner waits for the invocation of the current or extra-activity operation by the provider	Request invocation of the payment with the transfer of the data ccn
Reply operation *Rep-OP*	External	The partner service responds to a request for the execution of an operation previously accepted through a receive operation	Execution of the payment operation using the transferred ccn
End operation *End-OP*	External	The partner service informs the provider about the end of the operation execution	send invoice operation is over
Invoke security-action *Sec-A*	Internal/ External	Activation of the obligation by the provider or/and the partner service when the data retention time is over	Hide email
Other operation *other*	External/Internal	The provider and/or the partner can activate the operations which are not specified in the SLA. Such operations may or not use the private data	Statistic, maintenance process
Clock assignation activity		Assignment of clock variable for each triggered operation (activation time, end time) or for each collected data	

The *monitor* collects the raw information to be monitored regarding the monitoring private units from the different databases of the system. The usage stored in the temporal views, the monitoring private units stored in the requirement tables (Fig. 7) and the privacy violations are sent to the *containment checker* component in order to check the *usage compliance*.

Containment checker verifies if the recorded usages are compliant with the requirements being monitored. In cases where the recorded usages are not consistent with clauses in the SLA, the monitor records the violations in a violation database.

Usage flow view visualization. An intuitive graphical query language is provided to visualize the usage flow view abstracted from the PDUF. It allows to display both the results of query processing and checking of the containment process.

6.2 Implementation

We have implemented the proposed Privacy aware SLA Monitoring approach in a prototype system, called *PaM system*. A system demo. was conducted in the ICDE [33]. This prototype realizes the architecture of the framework that we have discussed in Sect. 6.1. In the prototype, we developed a simulator to provide the live back-end services calls. The current prototype runs for a single bilateral contract. The *PaM system* provides an efficient tool for tight surveillance for the purpose of dynamically observing the private data usage flow. It has been implemented as a set of visual interfaces and tools using Java builder X as the programming language, and MYSQL server as the database management system. We describe below the various interfaces.

6.2.1 Execution operations engine

It has been developed to set the collaboration between the different services by the activation of aforementioned operations. Figure 8 illustrates a snapshot of *PaM system*. It is shows the different operations that can be activated by the system, and the execution of the receive operation *(Rec-OP)* between the provider and the bank service (Table 1). The engine generates the log of the events during the execution process. This event log is fed into our framework in order to provide the runtime information that is necessary for monitoring. The events can be the data collection activity, the execution of the internal operation by the provider, or the activation of external operation through the exchanged messages between the partner services and the provider. Each operation has two clock variables, the activation time and the end time.

6.2.2 Visual editor

The event filter, which is related to the usage flow view, is used as a basis to query the information in the log in order to trace the relevant execution of the operations. The queries (UFVs) are written via a visual editor by selecting a subset of the canonical fields (constraints) from the log, and relating them with AND and/or OR relationships. When the constraint is selected, the editor populates the related drop down menu with all the corresponding values defined in the specification. This may then be used to choose the desired values as the constraint. The *Query process* executes the SQL query and displays the result in a table depicted in Fig. 9.

6.2.3 Query language

The results of the query processing with identification of the non-compliance are visualized via an intuitive graphical query language which is based on symbols. The attributes (monitoring units) of each filter event are converted into graphical symbols,

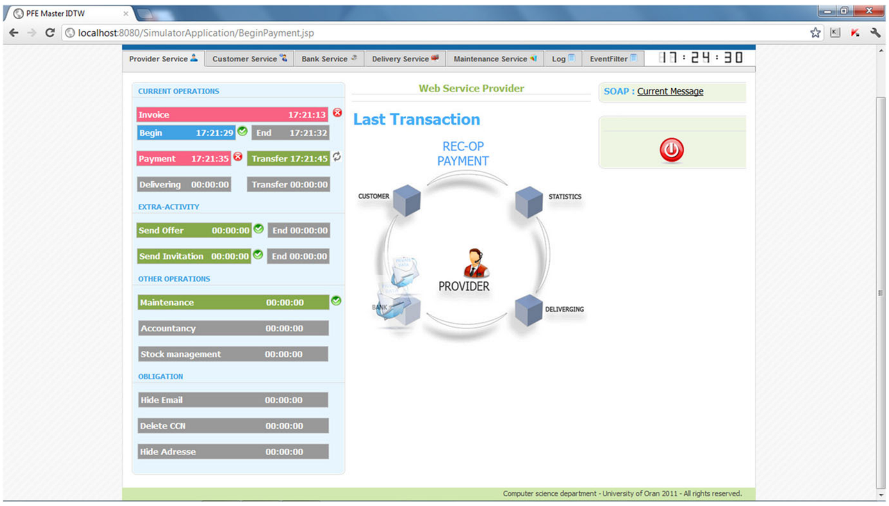
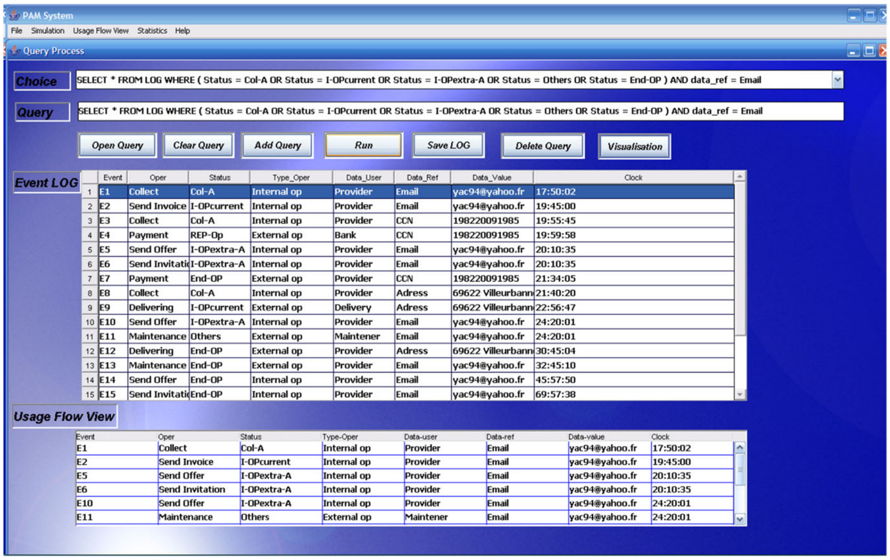


Fig. 8 Operations list of the Web service execution [the receive operation execution (Payment operation with the transfer of private data (ccn))]



Choice: SELECT * FROM LOG WHERE (Status = Col-A OR Status = I-OPcurrent OR Status = I-OPextra-A OR Status = Others OR Status = End-OP) AND data_ref = Email

Query: SELECT * FROM LOG WHERE (Status = Col-A OR Status = I-OPcurrent OR Status = I-OPextra-A OR Status = Others OR Status = End-OP) AND data_ref = Email

Event LOG

	Event	Oper	Status	Type_Oper	Data_User	Data_Ref	Data_Value	Clock
1	E1	Collect	Col-A	Internal op	Provider	Email	yac94@yahoo.fr	17:50:02
2	E2	Send Invoice	I-OPcurrent	Internal op	Provider	Email	yac94@yahoo.fr	19:45:00
3	E3	Collect	Col-A	Internal op	Provider	CCN	198220091985	19:55:45
4	E4	Payment	REP-Op	External op	Bank	CCN	198220091985	19:59:58
5	E5	Send Offer	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	20:10:35
6	E6	Send Invitati	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	20:10:35
7	E7	Payment	End-OP	External op	Provider	CCN	198220091985	21:34:05
8	E8	Collect	Col-A	Internal op	Provider	Adress	69622 Villeurbann	21:40:20
9	E9	Delivering	I-OPcurrent	External op	Delivery	Adress	69622 Villeurbann	22:56:47
10	E10	Send Offer	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	24:20:01
11	E11	Maintenance	Others	External op	Maintener	Email	yac94@yahoo.fr	24:20:01
12	E12	Delivering	End-OP	External op	Provider	Adress	69622 Villeurbann	30:45:04
13	E13	Maintenance	End-OP	External op	Provider	Email	yac94@yahoo.fr	32:45:10
14	E14	Send Offer	End-OP	Internal op	Provider	Email	yac94@yahoo.fr	45:57:50
15	E15	Send Invitati	End-OP	Internal op	Provider	Email	yac94@yahoo.fr	69:57:38

Usage Flow View

Event	Oper	Status	Type-Oper	Data-user	Data-ref	Data-value	Clock
E1	Collect	Col-A	Internal op	Provider	Email	yac94@yahoo.fr	17:50:02
E2	Send Invoice	I-OPcurrent	Internal op	Provider	Email	yac94@yahoo.fr	19:45:00
E5	Send Offer	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	20:10:35
E6	Send Invitation	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	20:10:35
E10	Send Offer	I-OPextra-A	Internal op	Provider	Email	yac94@yahoo.fr	24:20:01
E11	Maintenance	Others	External op	Maintener	Email	yac94@yahoo.fr	24:20:01

Fig. 9 Visual results of the query process

as shown in Fig. 10. The symbols form the basis of the query language. A sample illustration is provided in Fig. 11. The gray oval at the top of the usage flow view is the activation clause. Its attached white circle denotes the name of the private data, sky-blue box for the current, green box for the extra activity, red box for the other operations (violations), and yellow box for the obligation. Diamond represents the operation end. We also use two types of arrows to denote the order of the events. Thus,

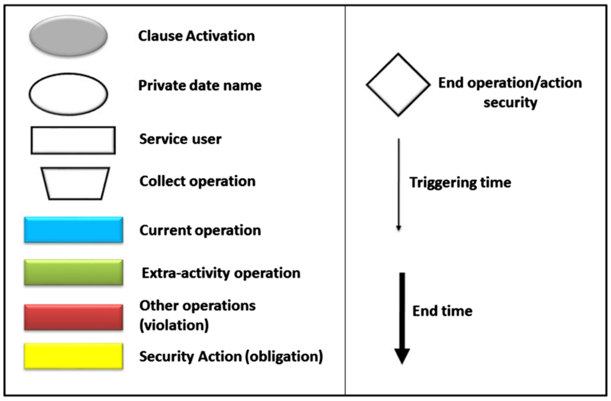
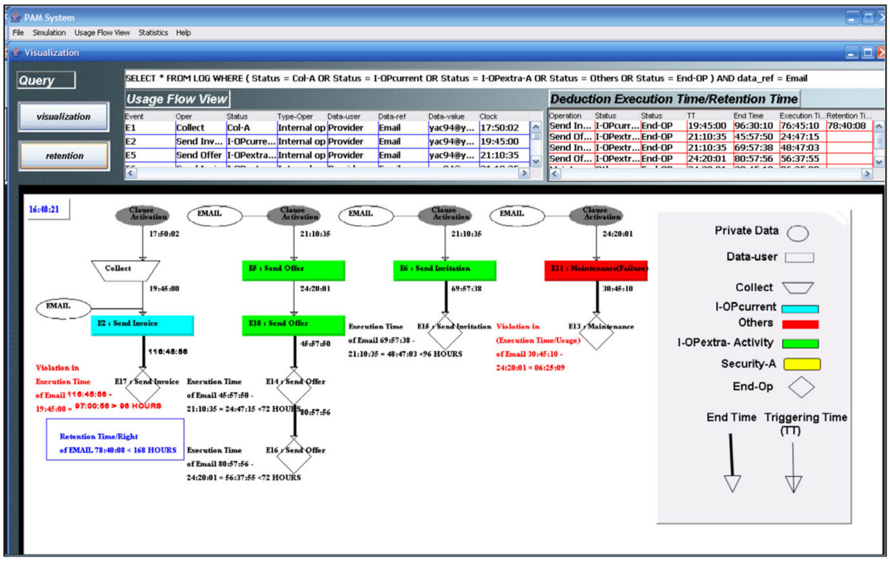


Fig. 10 An intuitive graphical query language



```
SELECT * FROM LOG WHERE ( Status = Col-A OR Status = I-OPcurrent OR Status = I-OPextra-A OR Status = Others OR Status = End-OP ) AND data_ref = Email
```

Usage Flow View

Event	Oper	Status	Type-Oper	Data-user	Data-ref	Data-value	Clock
E1	Collect	Col-A	Internal op	Provider	Email	yac94@y...	17:50:02
E2	Send Inv...	I-OPcurre...	Internal op	Provider	Email	yac94@y...	19:45:00
E5	Send Offer	I-OPextra...	Internal op	Provider	Email	yac94@y...	21:10:35

Deduction Execution Time/Retention Time

Operation	Status	Status	TT	End Time	Execution Ti...	Retention Ti...
Send In...	I-OPcurr...	End-OP	19:45:00	96:30:10	76:45:10	78:40:08
Send Of...	I-OPextr...	End-OP	21:10:35	45:57:50	24:47:15	
Send In...	I-OPextr...	End-OP	21:10:35	69:57:38	48:47:03	
Send Of...	I-OPextr...	End-OP	24:20:01	80:57:56	56:37:55	

Violation in Execution Time of Email 116:45:56 - 19:45:00 = 97:00:56 > 96 HOURS

Retention Time/Right of EMAIL 78:40:08 < 168 HOURS

Execution Time of Email 45:57:50 - 21:10:35 = 24:47:15 <72 HOURS

Execution Time of Email 80:57:56 - 24:20:01 = 56:37:55 <72 HOURS

Execution Time of Email 69:57:38 - 21:10:35 = 48:47:03 <96 HOURS

Violation in (Execution Time/Usage) of Email 30:45:10 - 24:20:01 = 06:25:09

Fig. 11 Usage flow view related to the execution time of each operation on email

the thin arrow indicates the triggering time and the bold arrow denotes the end time. As an example, Fig. 11 visualizes the result of the following query: *what are the execution times of each operation on the email?*. We can see every executed operation on the email with its corresponding triggering time and end time. It also shows two violations: (1) on the following requirements $\{R_1, R_4, R_5\}.OP$ (see Fig. 7) which require that the authorized operations are send invoice, send offer and send invitation. The corresponding notation in the graph is respectively [red box (E11:Maintenance(Failure))], (2) on the requirement $R1.T_e$ which requires that the execution time (T_e) of send invoice should always be equal or less than 4 days, the corresponding notation in the graph is respectively [Execution time violation E17: 116:45:56–E2: 19:45:00 = $97:00:56 >$96:00:00 h].

6.3 Experiments and results

In this section we report our experimental evaluation of the effectiveness of our PaM system. The experiments are conducted with two main objectives:

1. To measure the system efficiency in terms of detecting all operation violations through the usage frequency on different data type.
2. To study the SLA violation periods taking into account the usage frequency.

In our experiments, we implemented a *purchase service* (PS) involved by a client. This service requires a set of private data collected from the client including *email* to send the available products (offer) and possibly the invitations, credit card number *ccn* for the payment of invoice and *address* to deliver the products. We established and implemented an SLA on the system including all the clauses dealing with data right and data obligation. Our experiments were based on simulation of the Web service execution to generate synthetic data sets (Log, UFV, UDV, RDV). Next, we generated different usage flow paths related to all private data (email, ccn, address). Afterwards, we generated a set of events on each flow path incorporating random violations. These experiments highlight the service provider behavior on collected private data by displaying the usage compliance or violations (SLA breaches). The experiments are based on the analysis of usage views (UFV, UDV, RDV).

– *Operation compliance through the usage frequency*

In the first experiment, we investigate an important parameter, which is the usage frequency *oc* (occurrence) related to the operation *op* executed on data *d*. We intend to evaluate the effects of the usage frequency on the compliance (vs violation) of the operation for different types of data by varying the number of *oc*. The results of the experiment are presented in Fig. 12, where the x-axis is the operation *op* applied on a data *d* and y-axis is the value of the usage frequency *oc*. The blue point denotes the occurrences of each executed operation. The green one represents the occurrence requirement specified in the SLA and the red point is the violation. When the requirement meets the operation execution we have layered blue and green points. A variation of occurrences *oc* occurs on different operations for different types of data. We have one occurrence of *send invoice*, ten of *send offer*, five of *send invitation* and two of *maintenance task* on email, one of *payment* on ccn and five of *delivering* on address. The graph shows two usage violations (Fig. 13a): (*Maintenance task, the first activation at: 97:31:38, the second at: 318:38:18*).
From the experiment, we conclude that the usage frequency has an impact on the SLA compliance/violations that are monitored in the system. On one hand, some clauses can be as much triggered as possible for a type of data, so that they comply with the requirements such as the business need (e.g., send offer, send invitation). However, some clauses can be triggered in a small number of usage frequency. At the same time, the violation depends on the type of data. Consequently, we can suggest to assign weights for data type according to the degree of its sensitivity. For instance, credit card number is more sensitive than email. So, more vigilance is needed on a data type rather than on others. For that, we plan as a future work to provide a reasoning mechanism to handle the weights.

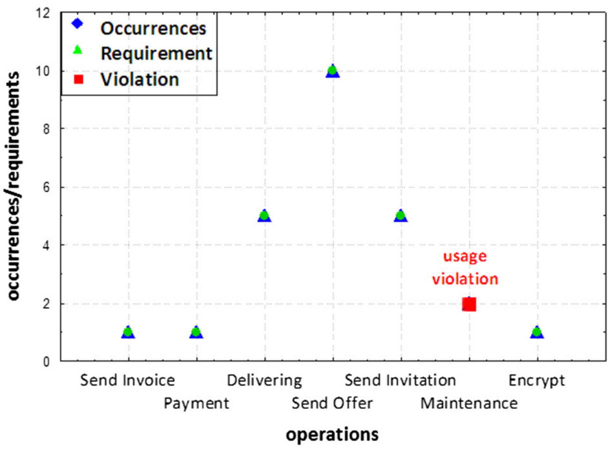


Fig. 12 The usage frequency on operations for different type of data (Compliance/Violation)

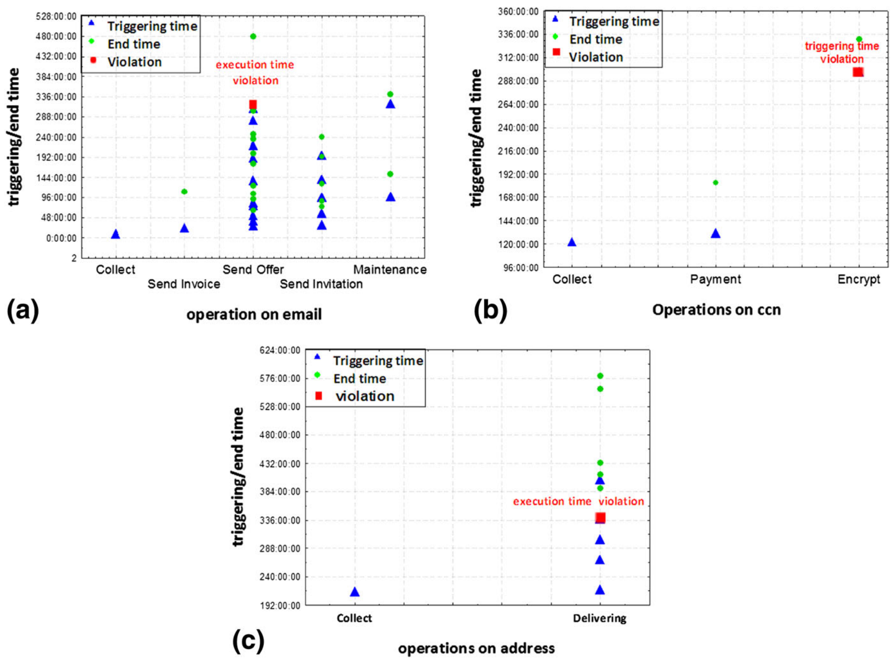


Fig. 13 The usage frequency on **a** email, **b** ccn, **c** address (Compliance, Violation)

– *Estimation of SLA violation periods for different types of data*

In this experiment, we investigate two important parameters. The first parameter is ϵ, operation triggering time, whereas the second parameter is β, the operation end time. We wish to estimate the SLA violation periods taking into account two view-points

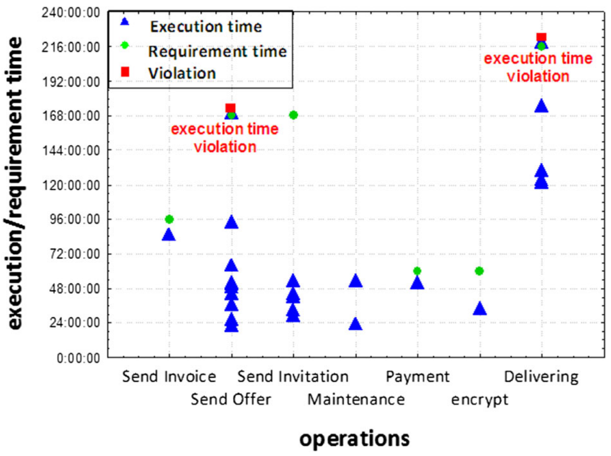


Fig. 14 The usage execution time (Compliance, Violation)

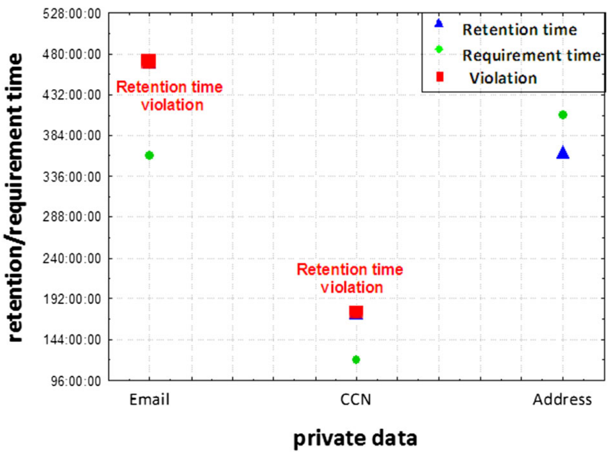


Fig. 15 The usage retention time (Compliance, Violation)

of time; operation execution time calculated from (ϵ,β) (Fig. 14) and retention time (Fig. 15). The results of the experiment are presented in Fig. 13, where the x-axis is the operation *op* applied on a data *d*, and y-axis is the value of an incremental time to be assigned to ϵ and β. The blue point denotes triggering time, the green one the end time and the red one is the violated time. Figure 13a–c display the usage frequency with its corresponding ϵ and β receptively on *email*, *ccn* and *address*. For instance, the execution of *send offer* on email has many occurrences and for this type of operation we do not care about ϵ,β since they are included in the ϵ of the obligation on the email (delete). However, there is a violation in the last occurrence due to the fact that ϵ is started after the obligation triggered time (delete). The payment execution time using *ccn* meets the SLA requirements ($\epsilon_1 = 130:51:38 \quad \beta_1 = 183:04:58$, $UDV.T_e.52:13:20 < R.T_e.60:00:00$), however, the time ϵ of the obligation operation *encrypt* is violated. Such violation has an impact on the retention time of *ccn* ($RDV.T_{ret}.175 : 20 : 00 > D.T_{ret}.120 : 00 : 00$). The execution of the delivering has occurred many times and we observe that there is a violation on execution time after three occurrences ($\epsilon_4 = 338 : 05 : 40, \beta_4 = 557 : 25 : 18, UDV.T_e.219 : 19 : 38 > R.T_e.216 : 00 : 00$), after that no violations have appeared. That fact is due for instance to a missing delivery because many delivering occurred before. From this experiment, we can conclude that the usage frequency has an impact on time compliance. Furthermore, for some types of operation, ϵ and β have no impact on the violation since they will not be over the ϵ of the obligations triggered time on data. The violations can also occur on time retention when an obligation is not triggered.

In summary, the evaluation confirms that the monitoring framework is able to detect the different types of violations at runtime effectively and efficiently. We conclude by stressing that the system does not stop the execution process when the violations occur and are detected, neither does it take corrective actions. However, it continues its running until the end of the contract. In our future work, we will investigate the extension of the prototype by considering the corrective actions for different violations.

7 Related work

Approaches dealing with the privacy compliance monitoring in Web services are very scarce in the literature. However, the problem of Web services and distributed business processes monitoring has received a lot of attention and many such systems have been developed. In our previous work we already provided a survey regarding the state-of-the-art in monitoring service-based applications [9]. In what follows, some of the works are presented [2–6,24,30,31,39,43,45]. The work in [31,45] focuses on monitoring of service-based software (SBS) systems specified in BPEL. They use event calculus for specifying the requirements that must be monitored. The runtime checking is done by an algorithm based on integrity constraint checking in temporal deductive databases. Barezi et al. [3,4] developed a tool that instruments the composition process of an SBS system in order to make it call external monitoring services that check assertions at runtime. The work in [2] is close to the previous works, the authors presented a novel approach to Web services described as BPEL processes. The approach offers a clear separation of the service business logic from the monitoring functionality. Moreover, it provides the ability to monitor both the behaviors of single instances of BPEL processes, as well as behaviors of a class of instances. In [42], the authors provided a monitoring language called BPath, which is an XPath-based language for both expressing and checking temporal and hybrid logical properties at run-time, making visibility on the execution of a business process by expressing and evaluating statistical queries over execution traces.

In [39], the authors proposed an approach to the automated synthesis and the run-time monitoring of Web service compositions. The automated synthesis rests on a set of existing component services that are modeled in the BPEL language, and a composition requirement. The latter expresses assumptions under which component services are supposed to participate in the composition, as well as conditions that the composition is expected to guarantee. Runtime monitoring matches the actual behaviors of the service

compositions against the assumptions expressed in the composition requirement, and reports violations. Beeri et al. [5] presented BP-Mon, a novel query language and system for monitoring BPs. BP-Mon offers a high level intuitive design of monitoring tasks. A novel optimization technique exploits available knowledge on the BP structure to speed up computation. In [30], the authors presented a framework to support the monitoring of service level agreements. The agreements that can be monitored are expressed in an extension of WS-Agreement.The main characteristic of the proposed extension is that it uses an event calculus based language, called EC-Assertion, for the specification of the service guarantee terms in a service level agreement that needs to be monitored at runtime. Lazovik et al. [28] proposed an approach based on operational and actor assertions which are used to express properties that must be true in one state before passing to the next, to express an invariant property that must be held throughout all the execution states, and to express properties on the evolution of process variables. While providing facilities for the verification of processes, these approaches do not tackle privacy issues into account.
In terms of privacy compliance, there exist few works including [10,26,34–36,38,47–49,52,53]. In [52], the authors examined privacy legislation to derive requirements for privacy policy compliance systems. They proposed an architecture for a privacy policy compliance system that satisfies the requirements and discussed the strengths and weaknesses of their proposed architecture. In [38], the authors introduced the concept of 'information transfer registry' as a mechanism to track compliance in a business to business network. The registry stores signed contracts specifying the consent an individual has given for information transfers for specific business purposes. Organizations register all information transfers from individual to business or business to business against the appropriate contract to document their compliance. In [49] the author proposed a graphical visual notation to facilitate the identification of private information vulnerabilities that can lead to privacy legislation non-compliance. In [34], the authors automated the management and enforcement of privacy policies (including privacy obligations) and the process of checking that such policies and legislation indeed comply with. The authors in [36] develop a Policy Compliance Checking System which models privacy compliance constraints, automates the assessment of the extent to which a particular computing environment is compliant and generates dashboard-style reports that highlight policy failures. The two last works are related to enterprise. While the previous works provide tools for privacy compliance, these approaches do not take private data usage flow into account and no formal method along with reasoning and also no time-related properties are discussed. Thus, in [10], we addressed the problem of run-time monitoring of compliance of the privacy agreement defining the users privacy rights and their possible handling by the service provider. A state machine based model is proposed to describe the PDUF toward monitoring which can be used by privacy analyst to observe the flow and capture privacy violations that may lead to non-compliance. The formalism adopted for the representation of the privacy units and misuses relies on linear temporal logic.
In order to provide better protection for personal data, the authors in [44] proposed PRMF, a privacy rights management framework which enforces personal data processing compliance with privacy policies related to organizational, legislative, and regulatory needs. PRMF can satisfy many aspects of privacy legislation, including security,

transparent processing, lawful basis, and finality—purpose limitation. The work in [53], described an approach for quantitatively assessing the likelihood that an organization will comply with privacy policy. The estimates allow organizations to be challenged if their likelihood to comply is perceived to be inadequate. Also, they allow customers to choose organization with high likelihoods of compliance. The works in [47,48] proposed an approach for compliance checking of agreed privacy policies and preferences in a federated identity management context. They introduce mechanisms and algorithms for policy compliance checking between federated service providers, on the basis of an innovative policy subsumption approach. The work in [26] focused on the private data discovery. Their objective is to develop ways to extract private data efficiently and effectively from unstructured and semi-structured content so as not to interfere with work activities. The private data may emerge from any type of computer-based activity, whether it is collaborative or not.

Recently, there has been research focusing on Compliance with privacy legislation for health care institutions [21,41]. Ghanavati et al. [21], describes a requirement management framework that enables health information custodians (HIC) to document and track compliance with privacy legislation as the legislation and hospital business processes evolve. A meta-model is defined to specify compliance tracking links between separate User Requirements Notation models of the HIC and privacy legislation. In [41], the authors have provided high level policies interpreted from European and national data protection law as privacy requirements for data disclosure and have captured similarity and possible conflict between the different frameworks across Europe through the use of SWRL rules, JESS and the protégé API. They have also specified by means of an ontology the concept of Enforceable Privacy Policy (Conforming to the XACML Standard).

In terms of privacy analysis, there exist few works including [50,51]. Yee [50] proposed a straightforward method for visual analysis of privacy risks in Web services, focusing the analyst's attention at locations that hold personal information at one time or another. Yee [51] proposed a design approach that incorporates privacy risk analysis of UML diagrams to minimize privacy risks in the final design. The approach iterates between the risk analysis and design modifications to eliminate the risks until a design is obtained that is close to being risk free.

In terms of privacy preserving related to the issues of constrained Web services selection, the work in [46] presented a novel approach for private selection of Web services by providing a comprehensive approach for protecting users and service providers' confidential data that is exchanged during service selection.

More recently, a formal study of privacy issues specific to scientific workflows has been done in [15–17]. In [15], the authors proposed the use of provenance views for preserving the privacy of module functionality in a workflow. The proposed model seeks to identify the smallest amount of data that needs to be hidden so that the functionality of every module is kept private. In [16], the authors discuss how to integrate privacy guarantees in the design of provenance management systems for workflows. The authors in [17] focused on privacy-preserving management of provenance-aware workflow systems. Furthermore, our work addresses privacy data usage rather than module privacy.

In terms of the application-level auditing, Fabbri et al. [18] investigated the novel problem of automatically explaining individual log records (accesses) in an access log. The authors develop a framework for modeling explanations which is based on a fundamental observation that accesses in specific classes of databases (including EHRs) occur for a reason, and the reason can be inferred from data in the database. Query containment has received significant attention in database research community. It is one of the most fundamental reasoning services related to querying the data and knowledge representation systems. In [22,23,29,40], the problem of query containment and equivalence has been studied in the more generic problem setting of query rewriting and materialized view selection. The problem of answering and rewriting queries using views in description logic and its complexities has been addressed in [7,12]. In [14], the authors dealt with a form of containment called *view-based query containment* in two specific contexts: conjunctive queries in relational databases and regular path queries in semi-structured databases. The query containment and query answering under description logic constraints was addressed in [13]. They introduce DLRreg, an expressive language for specifying database schema and non-recursive Datalog queries, and have presented decidability and undecidability results of both the problem of checking query containment and answering queries expressed in the schema. The CQC method to check query containment and query containment under constraints for queries over databases with safe negation in both IDB and EDB subgoals and with or without built-in predicates was studied in [19,20]. In [1], the authors discussed the scalability challenges of query containment in large predicate caches and proposed algorithms for fast containment checking in such an environment. Our work differs in the sense that it focuses on the query private monitoring unit expression containment issues and deals with checking private usage compliance of service execution.

8 Conclusion

We proposed an effective and formal approach to observe and audit the privacy compliance of Web services at runtime. We emphasized private data usage flow monitoring of SLA clauses, which is an important issue, and has not been addressed so far. We made use of a state machine based approach, that allows to take into account the timed-related properties of privacy requirements and facilitates the identification of private information violations. The monitored units are extracted from the contract requirements. Our approach supports monitoring a set of identified violations that lead to the non-compliance. We defined the problem of usage compliance with regards to the requirements, to the problem of query containment. A prototype of the framework has been provided along with a set of experiments. Our ongoing work includes: (1) The development of reasoning facilities to provide a diagnosis of violations, (2) The development of tools along with metrics for enhancing the privacy aware SLA from the observations, and (3) extension of the model to theoretically handle the multi-party SLA.

References

1. Amiri, K., Park, S., Tewari, R., Padmanabhan, S.: Scalable template-based query containment checking for web semantic caches. In: Proceedings of the 19th International Conference on Data Engineering. ICDE 2003, pp. 493–504. IEEE Computer Society, Bangalore (2003)
2. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-time monitoring of instances and classes of web service compositions. In: Proceedings of the IEEE International Conference on Web Services. ICWS 2006, pp. 63–71. IEEE Computer Society, Chicago, Illinois (2006)
3. Baresi, L., Ghezzi, C., Guinea, S.: Smart monitors for composed services. In: Proceedings of the ICSOC 2004. Second International Conference on Service Oriented Computing, ICSOC 2004, pp. 193–202. ACM Press, New York (2004)
4. Baresi, L., Guinea, S.: Towards dynamic monitoring of ws-bpel processes. In: Proceedings of the Third International Conference on Service Oriented Computing, ICSOC 2005, pp. 269–282. Springer, Amsterdam (2005)
5. Beeri, C., Eyal, A., Milo, T., Pilberg, A.: Monitoring business processes with queries. In: Proceedings of the 33rd International Conference on Very Large Data Bases. VLDB 2007, pp. 603–614. ACM, University of Vienna (2007)
6. Beeri, C., Eyal, A., Milo, T.: A. Pilberg: Query-based monitoring of bpel business processes. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. SIGMOD 2007, pp. 1122–1124. ACM, Beijing (2007)
7. Beeri, C., Levy, A.Y., Rousset, M.C.: Rewriting queries using views in description logics. In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 99–108. ACM, Tucson (1997)
8. Benbernou, S., Brandic, I., Cappiello, C., Carro, M., Comuzzi, M., Kertész, A., Kritikos, K., Parkin, M., Pernici, B., Plebani, P.: Modeling and negotiating service quality. In: S-CUBE Book, pp. 157–208. Springer, Berlin (2010)
9. Benbernou, S., Cavallaro, L., Hacid, M.S., Kazhamiaki, R., Kecskemeti, G., Poizat, J., Silvestri, F., M. Uhlig, B.W.: State of the art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of SBAs. S-Cube Deliverable PO-JRA **1**(1) (2008)
10. Benbernou, S., Meziane, H., Hacid, M.S.: Run-time monitoring for privacy-agreement compliance. In: Proceedings of the Fifth International Conference on Service Oriented Computing, ICSOC 2007, pp. 353–364. Springer, Vienna (2007)
11. Benbernou, S., Meziane, H., Li, Y., Hacid, M.S.: A privacy agreement model for web services. In: Proceedings of the IEEE International Conference on Services Computing, SCC 2007, pp. 196–203. IEEE Computer Society, Salt Lake City (2007)
12. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Answering queries using views in description logics. In: Proceedings of the 6th International Workshop on Knowledge Representation meets Databases. KRDB 1999, pp. 6–10. CEUR-WS.org, Linkping (1999)
13. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Conjunctive query containment and answering under description logic constraints. TOCL **9**(3), 22 (2008)
14. Calvanese, D., Giacomo, G.D., Lenzerini, M., Vardi, M.Y.: View-based query containment. In: Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS 2003, pp. 56–67. ACM, San Diego (2003)
15. Davidson, S., Khanna, S., Milo, T., Panigrahi, D., Roy, S.: Provenance views for module privacy. In: Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. PODS 2011, pp. 175–186. ACM, Athens (2011)
16. Davidson, S.B., Khanna, S., Roy, S., Stoyanovich, J., Tannen, V., Chen, Y.: On provenance and privacy. In: Proceedings of the Database Theory. 14th International Conference, ICDT 2011, pp. 3–10. ACM, Uppsala (2011)
17. Davidson, S.B., Khanna, S., Tannen, V., Roy, S., Chen, Y., Milo, T., Stoyanovich, J.: Enabling privacy in provenance-aware workflow systems. In: Fifth Biennial Conference on Innovative Data Systems Research, CIDR 2011, pp. 215–218. Online Proceedings. www.crdrdb.org 2011, Asilomar, CA, USA (2011)
18. Fabbri, D., LeFevre, K.: Explanation-based auditing. In: Proceedings of the 38th International Conference on Very Large Data Bases VLDB2012, Istanbul, pp. 1–12 (2012)

19. Farre, C., Teniente, E., Urpi, T.: The constructive method for query containment checking. In: Proceedings of the 10th International Conference Database and Expert Systems Applications, DEXA 1999, pp. 583–593. Springer, Florence (1999)
20. Farre, C., Teniente, E., Urpi, T.: Checking query containment with the CQC method. Data Knowl. Eng. **53**(2), 163–223 (2005)
21. Ghanavati, S., Amyot, D., Peyton, L.: A requirements management framework for privacy compliance. In: Proceedings of the Workshop em Engenharia de Requisitos, WER 2007, Toronto, pp. 149–159 (2007)
22. Halevy, A.Y.: Theory of answering queries using views. SIGMOD Rec. **29**(4), 40–47 (2000)
23. Halevy, A.Y.: Answering queries using views: a survey. VLDB J. **10**(4), 270–294 (2001)
24. Kazhamiakin, R., Pandya, P., Pistore, M.: Representation, verification, and computation of timed properties in web. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2006, pp. 497–504. IEEE Computer Society, Chicago (2006)
25. Kolaitis, P., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 205–213. ACM, New York (1998)
26. Korba, L., Wang, Y., Geng, L., Song, R., Yee, G., Patrick, A.S., Buffett, S., Liu, H., You, Y.: Private data discovery for privacy compliance in collaborative environments. In: Proceedings of the 5th International Conference Cooperative Design. Visualization, and Engineering, CDVE 2008, Lecture Notes in Computer Science, vol. 5220, pp. 142–150. Springer, Calvià (2008)
27. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benbernou, S., Brandic, I., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. **46**(1), 1 (2013)
28. Lazovik, A., Aiello, M., Papazoglou, M.: Associating assertions with business processes and monitoring their execution. In: Proceedings of the Second International Conference Service-Oriented Computing—ICSOC 2004, pp. 94–104. ACM, New York (2004)
29. Levy, A.Y., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS 1995, pp. 95–104. ACM, San Jose (1995)
30. Mahbub, K., Spanoudakis, G.: Monitoring WS-agreement: an event calculus-based approach. In: Test and Analysis of Web Services, pp. 265–306. Springer, Berlin (2007)
31. Mahbub, K., Spanoudakis, G.: Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2005, pp. 257–265. IEEE Computer Society, Orlando (2005)
32. Meziane, H., Benbernou, S.: A dynamic privacy model for web services. Comput. Stand. Interfaces **32**(5–6), 288–304 (2010). Elsevier
33. Meziane, H., Benbernou, S., Zerdali, A., Hacid, M.S., Papazoglou, M.: A view-based monitoring for privacy-aware web services. In: Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, pp. 1129–1132. IEEE computer society, Long Beach (2010)
34. Mont, M.C., Pearson, S., Thyne, R.: A systematic approach to privacy enforcement and policy compliance checking in enterprises. In: Proceedings of the Third International Conference Trust and Privacy in Digital Business, TrustBus 2006, pp. 91–102. Springer, Krakow (2006)
35. Pearson, S., Allison, D.: A model-based privacy compliance checker. Int. J. E-Bus. Res. **5**(2), 63–83 (2009)
36. Pearson, S., Allison, D.: Privacy compliance checking using a model-based approach. E-Business Applications for Product Development and Competitive Growth: Emerging Technologies, IGI GLOBAL, pp. 199–220 (2011)
37. Pernici, B., Siadat, S.H., Benbernou, S., Ouziri, M.: A penalty-based approach for qos dissatisfaction using fuzzy rules. In: Proceedings of the 9th International Conference on Service-Oriented Computing, ICSOC 2011, pp. 574–581. Springer, Berlin (2011)
38. Peyton, L., Nozin, M.: Tracking privacy compliance in b2b networks. In: Proceedings of the 6th International Conference on Electronic Commerce, ICEC 2004, pp. 376–381. ACM, Delft (2004)
39. Pistore, M., Traverso, P.: Assumption-based composition and monitoring of web services. In: Test and Analysis of Web Services, pp. 307–335. Springer, Berlin (2007)
40. Pottinger, R., Halevy, A.Y.: Minicon: a scalable algorithm for answering queries using views. VLDB J. **10**(2–3), 182–198 (2001)
41. Rahmouni, H.B., Solomonides, T., Mont, M.C., Shiu, S.: Privacy compliance in european healthgrid domains: an ontology-based approach. In: Proceedings of the Twenty-Second IEEE International

Symposium on Computer-Based Medical Systems, CBMS 2009, pp. 1–8. IEEE Computer Society, Albuquerque (2009)

42. Sebahi, S., Hacid, M.S.: Business process monitoring with BPath. In: Proceedings of the International Conferences on On the Move to Meaningful Internet Systems: OTM 2010—Confederated, CoopIS, IS, DOA and ODBASE, pp. 446–453. Springer, Berlin (2010)
43. Simmonds, J., Gan, Y., Chechik, M., Nejati, S., O'Farrell, B., Litani, E., Waterhouse, J.: Runtime monitoring of web service conversations. IEEE Trans. Serv. Comput. **2**(3), 223–244 (2009)
44. Song, R., Korba, L., Yee, G.: Privacy rights management for privacy compliance systems. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications, AINA 2007, pp. 620–625. IEEE Computer Society, Niagara Falls (2007)
45. Spanoudakis, G., Mahbub, K.: Non-intrusive monitoring of service-based systems. IJCIS **15**(3), 325–358 (2006)
46. Squicciarini, A.C., Carminati, B., Karumanchi, S.: A privacy-preserving approach for web service selection and provisioning. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2011, pp. 33–40. IEEE Computer Society, Washington (2011)
47. Squicciarini, A.C., Casassa-Mont, M., Bertino, E., Bhargav-Spantzel, A.: Automatic compliance of privacy policies in federated digital identity management. Tech. Rep. HPL-2008-8, HP Laboratories Bristol (2008)
48. Squicciarini, A.C., Mont, M.C., Spantzel, A.B., Bertino, E.: Automatic compliance of privacy policies in federated digital identity management. In: Proceedings of the 9th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2008, pp. 89–92. IEEE Computer Society, New York (2008)
49. Yee, G.: Visualization for privacy compliance. In: Proceedings of the 3rd Workshop on Visualization for Computer Security, VizSEC 2006, pp. 117–122. ACM, Alexandria (2006)
50. Yee, G.: Visual analysis of privacy risks in web services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2007, pp. 671–678. IEEE Computer Society, Salt Lake City (2007)
51. Yee, G.O.M.: Towards designing e-services that protect privacy. Int. J. Secur. Softw. Eng. **1**(2), 18–34 (2010)
52. Yee, G., Korba, L.: Privacy policy compliance for web services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2004, pp. 158–165. IEEE Computer Society, San Diego (2004)
53. Yee, G., Korba, L., Song, R.: Assessing the likelihood of privacy policy compliance. In: Proceedings of the IFIP TC-11 23rd International Information Security Conference IFIP 20th World Computer Congress, pp. 723–727. Springer, Milano (2008)
54. Zemni, M.A., Benbernou, S., Carro, M.: A soft constraint-based approach to qos-aware service selection. In: Proceedings of the 8th International Conference on Service-Oriented Computing, ICSOC 2010, pp. 596–602. Springer, Berlin (2010)