A Data Assurance Policy Specification and Enforcement Framework for Outsourced Services

Jun Li, Bryan Stephenson, Hamid R Motahari-Nezhad, Sharad Singhal

HP Laboratories
HPL-2009-357

Keyword(s):

Security and Privacy in Services, Security and Privacy Management in Data Collection, Transformation and Dissemination, Service Oriented Computing, Software as Services, Services Delivery Platform and Methodology

Abstract:

Many cloud service providers offer outsourcing capabilities to businesses using the software-as-a-service delivery model. However, challenges remain for widespread acceptance of this delivery model as it requires business critical data to be stored and processed outside the control of the business. The ability to manage data in compliance with regulatory and corporate policies, which we refer to as data assurance, is an essential success factor for this delivery model. The challenges include the ability to express service data assurance capabilities, capture customers' requirements and finally enforce these policies inside service providers' environments. This paper addresses these challenges by proposing GEODAC (Global Enforcement Of Data Assurance Controls), a policy framework that enables the expression of both service providers' capabilities and customers' requirements, and enforcement of the agreed-upon requirements in service providers' environments. High-level policy statements are backed in the service provider environment with a state machine-based representation of policies with each state representing a data lifecycle stage. Data assurance related policies including data retention, data migration, and data appropriateness for use can be described in this framework and enforced correspondingly. The approach has been implemented in a prototype tool and evaluated in a distributed services environment.

External Posting Date: November 6, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: November 6, 2009 [Fulltext]

© Copyright 2009 Hewlett-Packard Development Company, L.P.

A Data Assurance Policy Specification and Enforcement Framework for Outsourced Services

Jun Li, Bryan Stephenson, Hamid R. Motahari-Nezhad, and Sharad Singhal

{jun.li, bryan.stephenson, hamid.motahari, sharad.singhal}@hp.com

Abstract—Many cloud service providers offer outsourcing capabilities to businesses using the software-as-a-service delivery model. However, challenges remain for widespread acceptance of this delivery model as it requires business critical data to be stored and processed outside the control of the business. The ability to manage data in compliance with regulatory and corporate policies, which we refer to as data assurance, is an essential success factor for this delivery model. The challenges include the ability to express service data assurance capabilities, capture customers' requirements and finally enforce these policies inside service providers' environments. This paper addresses these challenges by proposing GEODAC (*G*lobal *E*nforcement *O*f *D*ata *A*ssurance *C*ontrols), a policy framework that enables the expression of both service providers' capabilities and customers' requirements, and enforcement of the agreed-upon requirements in service providers' environments. High-level policy statements are backed in the service provider environment with a state machine-based representation of policies with each state representing a data lifecycle stage. Data assurance related policies including data retention, data migration, and data appropriateness for use can be described in this framework and enforced correspondingly. The approach has been implemented in a prototype tool and evaluated in a distributed services environment.

Index Terms—Security and Privacy in Services; Security and Privacy Management in Data Collection, Transformation and Dissemination; Service Oriented Computing; Software as Services; Services Delivery Platform and Methodology

1 INTRODUCTION

INFORMATION technology is in the midst of a large shift—one that is transforming how vendors offer software products and people use software, access information and store content. The advances in cloud computing enable offering of hardware infrastructure through shared resources, and on top of that the delivery of software-as-a-service (SaaS) over the cloud infrastructure is the latest software delivery mechanism [34]. Many businesses are eager to take advantage of these advances to run certain business functions in a more cost-effective manner by outsourcing them to cloud service providers.

While this delivery model provides many opportunities, it poses many challenges that hinder its wide adoption. It requires customers to share, store and process critical business data in the cloud, where they have little control today. Indeed, the ability to manage customer data in service provider environments in compliance with regulatory and corporate policies and in a transparent manner to service customers is identified as one of the main problems in SaaS adoption [19]. This problem is related to the issue of *information assurance*, which according to the U.S. National Information Assurance Glossary [38], refers to offering "measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and nonrepudiation." We use the term "data assurance" to describe the set of information assurance requirements and enforcement capabilities which provide the ability to manage data in compliance with regulatory and corporate policies. Examples of regulatory compliance include the Data Protection Directive under EU privacy law [9], HIPPA [14] and PCI DSS [26].

To alleviate data assurance concerns and enable the use of outsourced services with a well-understood level of risk, we need to support service providers and customers to express, communicate and enforce data assurance policies in outsourced service environments. The challenges of providing such support are as follows:

1- Characterizing compliance and regulatory requirements for data assurance such as data retention, data migration and data confidentiality in outsourced services environments is required. While many people have recognized the urgent need for data assurance mechanisms [10], these aspects have not received adequate attention in the literature, and it is difficult to express such requirements using existing policy languages.

2- Businesses are sensitive about sharing business critical data like customer lists, sales data, financial data, or product data. While data handling policies can be specified in service level agreements or contracts [17], these contracts are usually written in legal terms and do not contain fine grained policies necessary to manage data during operations. Even when the service provider agrees to fine-grained policies, it is possible for data to be mishandled inadvertently because multiple service providers are often involved in delivering any service [37].

Thus it is also important to develop policy assurance frameworks to ensure data is handled in service provider environments according to each customer's policies. The framework should enable binding different policies to different data object types and translating these requirements into machine interpretable and enforceable policy.

3- The selected data assurance policies have to be realized in the service provider environment. Currently, there is no systematic support or mechanism to enforce the customer's data assurance policies for data objects (e.g., customer list, credit card number, etc.) within the service providers' environments. Existing approaches use network-level techniques, which are not adequate [1, 30].

To address the above problems, we propose a policy modeling and enforcement framework which allows security experts and system designers to work together to define data assurance policies, configure such policies to accommodate the customer's specific requirements, and enforce the configured policies correspondingly in the services' runtime environment. In particular, we make the following unique and novel contributions in this paper:

1- We characterize the data assurance requirements in an outsourced services environment consisting of data retention, data privacy, data migration, etc.

2- We present WS-DataAssurancePolicy, by extending the WS-Policy framework, a policy specification language which allows service providers to express capabilities, and capture the requirements of service customers at the desired data granularity level.

3- We introduce the GEODAC (*G*lobal *E*nforcement *O*f *D*ata *A*ssurance *C*ontrols) policy framework that enables the expression of both service providers' capabilities and customers' requirements, and enforcement of the agreed-upon requirements in service providers' environments. For service customers, it enables security experts and system designers to work together to define sets of data assurance policies, and bind these policies to data elements used in the service, allowing different policy sets to govern different data elements. It automatically translates the specified capabilities and data bindings into the WS-DataAssurancePolicy language.

4- We propose a runtime enforcement mechanism for service providers to enforce WS-DataAssurancePolicy requirements. Policies are expressed in state machines with states representing lifecycle stages of the data. A state machine-based policy model allows enforceable actions on data to become observable and auditable, and also enables incorporating long-lasting and human-dependent actions in the policy.

5- We implement the approach in a prototype system that supports data retention, data migration and data confidentiality policies. We have evaluated the approach in a prototyped distributed services environment.

It should be noted that the current paper extends our earlier work on enforcing data assurance policies in service providers' environments [18]. In particular, the novel contributions of this paper include items 1 (characterization of requirements), 2 (proposing WS-DataAssurancePolicy) and 3 (GEODAC framework for policy customization and binding) above, and also extending the runtime environments to map policies expressed in WS-DataAssurancePolicy into implementable policies expressed as state machines.

The remainder of this paper is structured as follows. In section 2 we present an example use case and define the data assurance requirements. Section 3 describes the GEODAC framework. In Section 4 we introduce the WS-DataAssurancePolicy language, and in Section 5 we describe a data assurance console which customers use to configure data assurance capabilities without needing to author policies directly in this language. In Section 6 we describe our state machine-based enforcement mechanism for these policies and in Section 7 we describe the policy enforcement platform architecture and its implementation. Section 8 presents related work. We conclude and present our future plans in Section 9.

2 Data Assurance Requirements

In this section, we first introduce a use case scenario that heavily involves service outsourcing and then use it to characterize key data assurance requirements in outsourced services environments.

2.1 Marketing Campaign Use Case Scenario

To more clearly illustrate the problem and our proposed solution, we examine a product marketing campaign run by a fictional company called Nullco.

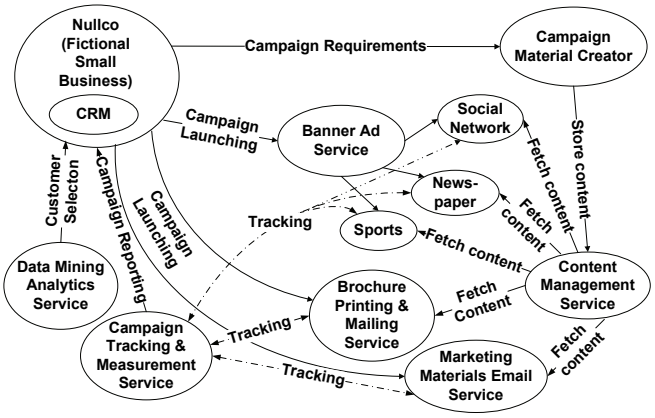


Fig. 1. Services composed for on-line marketing.

Shown in Fig. 1, the marketing campaign relies on services from ten different providers. A data mining analytics service identifies the targets for the campaign, based on Nullco's historical customer purchase information. A creative agency designs campaign materials, which are hosted at a content management service. The campaign is launched through multiple channels that include direct mail, email, and banner ads. The direct mail and email services require Nullco to provide them customer contact information. The banner ad service subcontracts its work to other ad providers specializing in social networking, newspaper, and sports. A tracking service gauges effectiveness and fine-tunes the running campaign. Finally, leads generated are stored in Nullco's CRM system and are used to evaluate the effectiveness of the campaign.

2.2 Requirements for Data Assurance

There are many data assurance requirements which must be considered in engagement with outsourced services. In this section, we list the important data assurance requirement categories, identify requirements within each category, and define the necessary control actions for each requirement. A control action specifies concrete activities that must be performed to enforce the requirement. The list is by no means exhaustive. We recognize the related and important requirement of access control, which has been investigated thoroughly (e.g., [6][7][43]), but it is not discussed in this paper. Table 1 summarizes all the data assurance categories, requirements and associated control actions we identified, which we describe in the following.

2.2.1 Privacy Requirements

Geographical location controls: Various privacy laws limit the locations where data can be stored or processed. For example, any PII (personally-identifiable information) of EU citizens needs to remain in the EU or in entities which satisfy the Safe Harbor principles [32]. Nullco sends PII to various services during the marketing campaign, and thus is responsible to ensure compliance with privacy laws. A control action is needed to restrict data location to certain jurisdictions such as the EU.

Safe Harbor verification: Using companies which certify compliance with the Safe Harbor principles is another means to address certain EU Privacy Directive [9] requirements. A control action is needed to restrict selection of services to those which certify Safe Harbor compliance.

Data breach notification: Many countries and US states have data breach notification requirements for PII which is reasonably believed to have been accessed by unauthorized parties. Thus to meet its notification requirements under these laws, a control action is needed to ensure that Nullco is informed of any suspected breaches of their data at any subcontractor.

2.2.2 Data migration requirements

The outsourcer, such as the banner ad service in the example, may migrate data to subcontractors, which can be a source of data breaches [37]. This doesn't absolve the data owner, Nullco, of legal responsibility for this data. To address this risk, four control actions are needed.

Pre-migration checks: Nullco may have various requirements which govern the selection of services in addition to the Safe Harbor verification as discussed above. Reputation services may be consulted to aid in selection. Nullco may require each service provider to have been in operation for a period of time. Nullco may even wish to pre-approve each migration of data. In general, a method is needed to specify and enforce a variety of pre-migration checks to ensure the data will be migrated to other services only in compliance with policy.

Data fingerprinting: To reduce the likelihood of intentional data theft and enable tracing of data leaks, Nullco may wish to fingerprint a data set such as its customer list before sending it to service providers [39]. By providing each service with a uniquely-marked copy of the data, it is possible to trace a data leak to the service which received that copy of the data.

Notification of data migration: Nullco may wish to know which subcontractors receive a copy of certain data sets. A control action is needed to notify Nullco when data is migrated to a subcontractor, such as the banner ad subcontractor specializing in sports-related advertising.

Propagation of Policies: When data is sent to subcontractors, Nullco requires the data assurance policies to be propagated to and enforced by all companies using the data. A control action is needed to propagate the policies.

2.2.3 Data retention requirements

Nullco's confidential information is sent to many service providers including direct contractors and subcontractors. Nullco wishes to place limits on how long data can be retained by the various parties to reduce the likelihood of data breach and the resulting costly notification process. Two control actions are needed.

Data deletion: Data must be deleted after a particular time or event, such as when the data is no longer needed to fulfill the service.

Notification of data deletion: Nullco wants to know that certain data has actually been deleted. This control action notifies someone that the data has been destroyed.

2.2.4 Data confidentiality

Nullco needs to ensure that the campaign materials and other sensitive data remain confidential in different stages of the data lifecycle including creation, sharing, and archiving. Access control is a necessary means to help ensure data confidentiality. In addition to access control, two control actions are needed.

Encrypt in storage: This control action requires data to be encrypted at rest, which greatly reduces the number of people who need to be trusted in order to preserve data confidentiality by preventing certain types of leaks [37].

Encrypt in transmission: This control action requires the data to be encrypted during transmission over any network link, which also reduces the number of people who need to be trusted in order to preserve data confidentiality by preventing certain other types of leaks.

2.2.5 Data availability

Nullco needs to ensure that the campaign materials hosted at the content management service are available to service providers such as the email marketing service. While data availability is a key aspect of data assurance, it is typically addressed during SLA negotiations when planning the service engagement and delivered by selecting appropriate hosting infrastructure and data replication facilities within the service. Thus, our framework does not directly enforce this requirement, but can monitor availability (see Section 6.3.2).

2.2.6 Data integrity

If a service handles data which cannot tolerate errors, data integrity checks are required when interacting with that service and within that service to guard against and detect corrupt data. One of the needed control actions is to use a message authentication code for all transmission and storage of error-intolerant data.

Requirement	Control Action(s)	Control Label(s) in Policy Assertion (see Section 4)
Privacy Requirements		
Geographical location controls	Restrict data location to specified places	*RestrictLocation*
Safe Harbor verification	Verify data receiver is Safe Harbor certified	*CertifySafeHarbor*
Data breach notification	Notify upon suspected data breach	*DataBreachNotification*
Data Migration Requirements		
Pre-migration checks	Enable a range of checks before migration	*ReputationCheck, Longevity*
Data fingerprinting	Uniquely fingerprint data to trace leaks	*FingerprintData*
Notification of data migration	Notify that data has migrated successfully	*EmailNotification*
Propagation of Policies	Send all policies to all data receivers	*PropagatePolicies*
Data retention requirements		
Data deletion	Destroy data at the agreed-upon time/event	*DeleteBy*
Notification of data deletion	Notify that data has been destroyed	*EmailNotification*
Data Confidentiality Requirements		
Encrypt in storage	Encrypt data in any storage medium	*EncryptStorage*
Encrypt in transmission	Encrypt data sent over any network link	*EncryptTransmission*
Data Integrity Requirements		
Guard against corrupt data	Verify data integrity upon each retrieval	*CheckIntegrity*
Usage Appropriateness Requirements	Enable approvals of actions and spot-checks	*RequireApproval*

Table 1. Requirements, control actions and control labels for data assurance.

2.2.7 Appropriateness for use

As a larger percentage of a corporation's data is generated and maintained by outsourcers as shown in this example, it becomes more important to monitor the quality of the data to ensure that it is appropriate for the uses of the data. For example, Nullco would like to have the marketing department approve campaign materials before usage to ensure they meet Nullco's brand standards. In the general case, spot-checks of data quality by both software and humans are needed. Consequently, data assurance processes need to support human decisions.

3 THE GEODAC FRAMEWORK: OVERVIEW

We present the GEODAC framework to be used by service providers and customers to express, specify and communicate their data assurance capabilities and policy requirements, and for service providers to enforce the policies in their environment.

Fig. 2 shows the overall GEODAC framework in a schematic way. In this approach, the service provider publishes its capabilities in terms of data assurance policies on individual data object types. The capabilities are described as parametric statements with acceptable values for each parameter. For example, the marketing email service may offer data retention capability for marketing collateral. The related statement identifies the data retention policy with a specified parameter for the expiry date when data will be destroyed.

The assumption is that the service provider has published an explicit definition of the data which is exchanged with the customer. For example, if the service has a WSDL interface then the incorporated XML schema definition serves this purpose. If a service has no WSDL interface then an explicit declaration of what data is stored by the service provider is needed, e.g., a data schema or the list of file names containing data has to be provided. This enables policy specification for storage services like Amazon Simple Storage Service (S3) [1].

At the customer side, the security expert uses a policy assurance console to retrieve the capabilities of a service like the marketing email service. The console provides the list of capabilities of the service for each data object type. This allows the security expert and system designer to cooperate to select desired capabilities and configure them in accordance to regulations and corporate policy.

Assume that the company has a records retention policy which states that service providers need to destroy their copies of any PII within 30 days after completing the service. There are two aspects of data assurance policies to be considered: data object identification and policy configuration. The former refers to selecting data object types that are considered PII. The later refers to choosing the appropriate protections on each data object type.

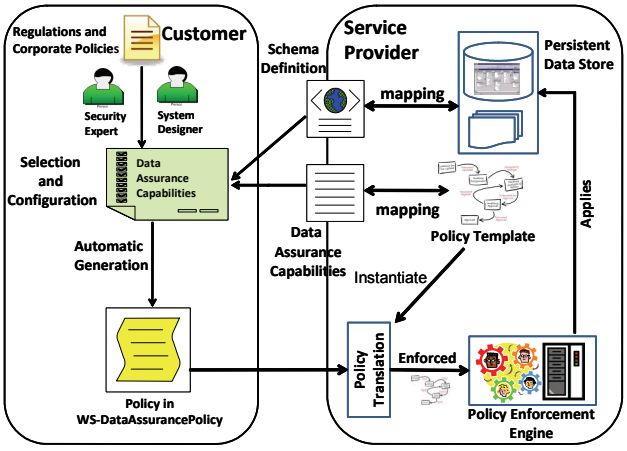


Fig. 2. The GEODAC framework for data assurance policy specification and enforcement.

During the process of policy configuration, a set of policies and their control actions are specified which include the details of data types and associated policy configurations. The console allows users to automatically generate the policy in the WS-DataAssurancePolicy language (described in Section 4).

The WS-DataAssurancePolicy text is then communicated with the service provider and acts as a contract between the service provider and the customer. The policy

enforcement engine in the service provider picks up the contracts, binds the published data types into concrete persistent data object types, and associates each requirement with a state machine-based policy specification. The state machine-based policy specification represents all the lifecycle stages of data in the environment of the service provider. The enforcement engine uses the mapping of published data schema into persistent store schema to enforce the policies at different levels of data granularity, e.g., database, table, record, directory and file levels.

The policy enforcement engine is built on top of a workflow engine to enable monitoring and tracking the lifecycle stages of data. Note that many data assurance policies require human involvement in decision making. For example, people may review and approve campaign material before it is served by the banner ad service. Therefore, an additional benefit of using a workflow engine is that it enables involving humans in the data lifecycle management and policy enforcement processes.

4 Data Assurance Policy Specification Language

In this section, we introduce our data assurance policy specification language. It allows service providers to present their data assurance capabilities to the service customers, as well as service customers to express their data assurance requirements. A service customer chooses the service and customizes its own data assurance policy based on the service capabilities. The data assurance policies are represented in XML format. We extend the WS-Policy language [2] to accommodate data assurance policies and call the policy language the *WS-DataAssurancePolicy* language.

4.1 Data Assurance Policy Specification

The design of our data assurance policy specification language is guided by the data assurance requirements identified in Section 2. The language is focused on data assurance for service outsourcing environments. The specification language supports *data assurance policy categories* that include data retention, data migration, data confidentiality, etc. Within each data assurance policy category, we can further specify the supported *data assurance policy assertions*. A policy assertion, in WS-Policy terms, identifies a behavior that is a requirement (or capability) of a policy subject [2]. For example, with respect to control actions of data retention, we define a *DeleteBy* policy assertion to denote at what absolute time the data needs to be deleted, *DeleteByAfterEngagement* to denote at what relative time after the engagement is finished, and *DeleteByAfterUpdate* to denote at what relative time after the last update on the data. Similarly, for data privacy, we define policy assertions of *DataBreachNotification, CertifySafeHarbor,* and *RestrictLocation* (See Table 1).

Our framework handles two data representations: the *transient data* exchanged between the service provider and the client, and the *persistent data* stored and processed in the service environment and potentially further transferred to other service environments. The client works with the schema definition of transient data, e.g., XML element type declared in WSDL. Our policy language incorporates the data type definition for both transient and persistent data types, and policies defined on them.

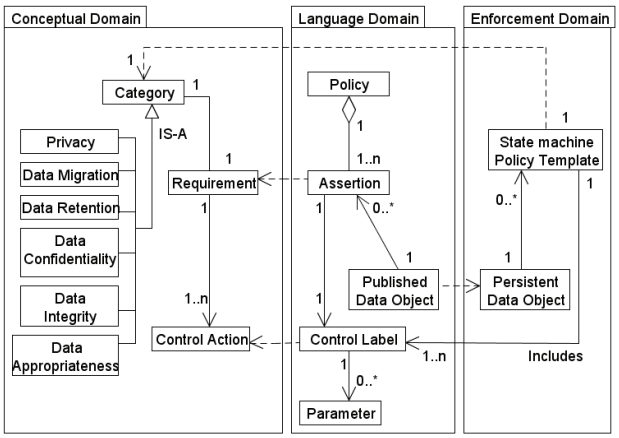


Fig. 3. Metamodel of WS-DataAssurancePolicy language.

Fig. 3 shows the metamodel of the WS-DataAssurancePolicy language. A policy is a set of assertions. An assertion has one control label, which is associated to a control action. An assertion formally represents a requirement in the language. A published data type can have one or more assertions defined and represents an associated persistent data type. We describe the language in detail in the following.

4.1.1 Data Object Definition

A data object definition in our policy specification consists of the object name, an optional textual description about the object, its data type definition of the exchanged data, and its data type definition when the data is in the persistent data stores of service providers or subcontractors.

An example of data object definition is shown in Fig. 4 for *CustomerPurchaseHistory*. The type definition for the exchanged data is encoded as an XML element *TransientDataRef* with a URL pointing to the corresponding data

```
<DataObjectDefinitions service=
      "http://www.superacim.com/DataMining">
  <DataObjectDefinition>
    <CustomerPurchaseHistory>
      <Description>data about customer purchase
transaction history</Description>
      <TransientDataRef published="true">
         http://www.superacim.com/DataMining/
         service.wsdl?object=
         CustomerPurchaseHistory
      </TransientDataRef>
      <PersistentDataRef published="false">
         http://www.superacim.com/DataMining/
         schema?data=CustomerPurchaseHistory
      </PersistentDataRef>
    </CustomerPurchaseHistory>
  </DataObjectDefinition>
  ….

</DataObjectDefinitions>
```

Fig. 4. Data object definition – example.

definition in the involved service's WSDL. The persistent data type definition is encoded as an XML element *PersistentDataRef* with a URL pointing to the corresponding backend schema definition. Such control of the persistent data definition exposure is encoded by an XML attribute called *published* in Fig. 4.

The mapping of published data types to persistent data objects can be cumbersomely complex. Therefore, in our language we allow the *PersistentDataRef* reference for mapping of published data types to persistent data types. This can be set by and is only visible to the service provider. In general, the backend service implementation may adopt high-level data mapping frameworks such as Hibernate [13] or a high-level business objects programming environment such as salesforce.com [31], in which a transient data object can have a one-to-one mapping to a persistent data object. This is because the entire service development starts with data object definition, and web services designed to manipulate such backend business objects have their interface definitions strictly follow what has been already defined in the business objects.

4.1.2 Service Data Assurance Capabilities

A service's data assurance capabilities are represented by the data assurance capabilities on individual data objects in the service's published data types. To specify its data assurance capabilities, each data object type will define all of the data assurance policy categories which the object type supports. In each category, all the policy assertions which the object type supports are defined.

```
<ServiceCapabilities service=
    "http://www.superacim.com/DataMining">
  <CustomerPurchaseHistory>
    <DataAssuranceCapabilities>
      <DataRetention template-url=
      "http://www.superacim.com/DataMining/
              template?data=
      CustomerPurchaseHistory?category=
            DataRetention">
        <DeleteBy/>
        <DeleteAfterLastUpdate/>
      </DataRetention>
      <DataMigration>
          .........
      </DataMigration>
    </DataAssuranceCapabilities>
  </CustomerPurchaseHistory>
  <CustomerProfileInformation>
    <DataAssuranceCapabilities>
      <DataRetention template-url=
         "http://www.superacim.com/DataMining/
         template?data=
      CustomerProfileInformation
           ?category=DataRention">
        <DeleteBy/>
        <DeleteAfterLastUpdate/>
      </DataRetention>
      <DataMigration>
          .........
      </DataMigration>
    </DataAssuranceCapabilities>
  </CustomerProfileInformation>
</ServiceCapabilities>
```

Fig. 5. Specification of data assurance service capabilities on persistent data objects.

Fig. 5 shows data assurance capability specification for the service *DataMiningAnalytics* for data objects *CustomerPurchaseHistory* and *CustomerProfileInformation*. For each object, the supported data assurance policy categories include data retention and data migration.

The supported policy assertions for the data retention capability for both objects in this example are *DeleteBy* and *DeleteAfterLastUpdate*. Note that a policy assertion can have configurable parameters for the service customer. Each policy assertion usually comes with a well-defined set of the configurable parameters derived from the corresponding runtime enforcement mechanism that is implemented by the service provider. Whether a policy assertion is configurable is specified in the capability, and configuration parameters along with acceptable values are given in the configuration schema. A unique URL is provided at the data assurance policy category level to specify the location of the configuration schema (if it is a separate XML file) for the corresponding data object. For example, a *DeleteBy* policy assertion specifies the acceptable expiration time formats, either in the format of {year, month, day}, or in the format of the number of milliseconds since a well-defined epoch time (e.g., Jan. 1, 1970, defined in Unix). According to the configuration template, an email address can be also provided to enable email notification after data deletion is performed.

Data assurance policy categories are independent of each other. Within the same data assurance policy category, certain policy assertions can be chosen independently from the others, and some will be chosen in a mutually exclusive manner. For example, for data retention, *DeleteBy* and *DeleteAfterLastUpdate* are mutually exclusive. The default is that policy assertions are independent from each other. If assertions are mutually exclusive, or the service provider's policy enforcement only supports a limited set of combinations of them, these limitations are specified following what is proposed in WS-Policy [2].

4.1.3 Service Customer Data Assurance Policy

Following the GEODAC framework presented in Section 3, once the service customer decides to use the service, a data assurance policy specification is provided by the service customer detailing requirements on how each data object type should be managed.

The service customer's data assurance policy is expressed using the same language described in Section 4.1.2. The set of policy assertions reflects the choices of service customers and does not necessarily include all supported capabilities for data objects, nor all of the supported policy assertions in each policy category. In each chosen policy assertion, the configuration of parameters according to the schema of data object assertion makes the policy assertion concrete.

Fig. 6 shows the involved data retention policies related to Fig. 5. There are two data objects involved, *CustomerPurchaseHistory* and *CustomerProfileInformation*. Each data object has its data reference pointing to the corresponding published data definition (shown in Fig. 4). Each policy assertion in each data object has its own configured data retention policy specification. More specifically, *CustomerPurchaseHistory* has its data retention policy specified by an absolute date in the future, denoted by a policy assertion of *DeleteBy*, and C*ustomerProfileInforma-*

```
<ServiceCustomerPolicies customer=
"http://www.nullco.com/MarketingCampaign">
  <CustomerPurchaseHistory dataRef=
      "http://www.superacim.com/DataMining/
      service.wsdl?object=
      CustomerPurchaseHistory">
    <DataAssurancePolicies>
      <DataRetention>
        <DeleteBy>
          <Date format="YEAR-MON-DAY">
            2012-05-15
          </Date>
        </DeleteBy>
      </DataRetention>
      <DataMigration>
         .............
      </DataMigration>
    </DataAssurancePolicies>
  </CustomerPurchaseHistory>
  <CustomerProfileInformation dataRef=
        "http://www.superacim.com/DataMining
        /service.wsdl?object=
        CustomerProfileInformation">
    <DataAssurancePolicies>
      <DataRetention>
        <DeleteAfterLastUpdate>
          <Duration format="Number Of Days">
            90
          </Duration>
          <EmailNotification>
            johndoe@nullco.com
          </EmailNotification>
        </DeleteAfterLastUpdate>
      </DataRetention>
      <DataMigration>
         .............
      </DataMigration>
    </DataAssurancePolicies>
  </CustomerProfileInformation>
</ServiceCustomerPolicies>
```

Fig. 6. Customer-specific data assurance requirements.

tion has its retention policy specified as a future date from last update, denoted by a policy assertion of *DeleteAfterLastUpdate*. Furthermore, because *CustomerProfileInformation* is determined to be more sensitive by Nullco, an email notification for deletion is specified by Nullco.

4.2 Policy Specification Extensibility

Our XML-based data assurance policy language is extensible by service providers to present a broad range of their data assurance capabilities beyond what we identified in Section 2. WS-DataAssurancePolicy language is XML-based and is backed with an XML schema, where all known assertions and acceptable configuration parameter values are defined. The extensions to WS-DataAssurancePolicy have to be defined in the schema, which have to be made available to service customers by the service provider along with the capabilities. This enables service customers to understand the structure and semantics of new capabilities offered by a particular service provider.

Not only can new capabilities be added to the language, but also the definition of existing capabilities can be customized and extended by a service provider. In particular, customization can be applied on the set of configurable parameters of each such assurance capability. For example, email notification of data deletion might be a specific add-on. It may become part of the configuration parameters of data retention capability for the service provider *SuperAcim*, but not for another service provider with no such feature. Such service provider-specific choices are embodied in the policy schema definition.

5 Assisting Customers in Selection and Configuration of Data Assurance Policies

We provide an interactive environment, called data assurance console, to facilitate the job of service customers to understand service provider capabilities and specify their data assurance requirements. This console reads the service provider's capability specifications and transforms them using Google Web Toolkit to generate a dynamic web UI. The console provides two complementary views: *data view* and *policy view*.

The data view shows the list of all published data object definitions for which data assurance capabilities are offered. This view allows selection by the system designer and security expert of the data object types which require protection in the service provider environment. The data object type names are declared in the published data schema by the service provider. Additionally, the data object may have descriptions (e.g., this data item is considered as PII), in the service capability definition, that will be displayed in the data view to simplify the selection process. For the above example, the system designer and security expert may select the data object *CustomerProfileInformation*, which is a complex XML element type in the schema definition of the email marketing service, because it is PII and thus requires certain protections.

The policy view shows the data assurance capabilities for each selected data object type. This view is used by the security expert to configure the requirements on the data objects of this type. Fig. 7 shows a screenshot of the editor showing the policy view for the *CustomerPurchaseHistory* data object type. The security expert may select data retention related control actions from this list. According to the template provided by the service provider, this requirement needs two configuration choices for the Delete Data capability: parameter type (one of "this many days after last update" or "on this date"), and its corresponding parameter value.

The console enables security experts to use an intuitive GUI to select and configure data assurance policies. At the end of the selection process, the console enables generating a policy specification in WS-DataAssurancePolicy

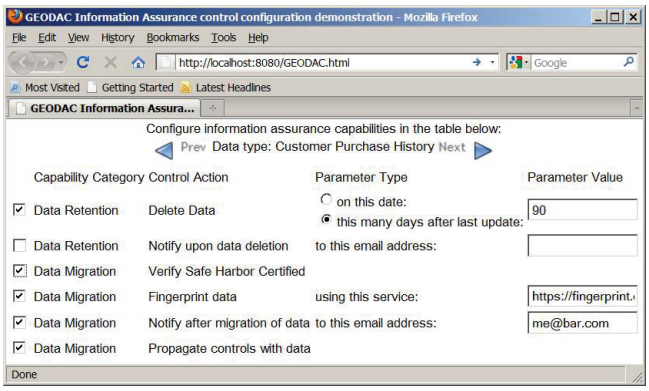


Fig. 7. Policy view of data assurance console.

covering all selected and configured policies. This document is communicated to the service provider for enforcement.

6 Enforcing Data Assurance Policies

The service customer's WS-DataAssurancePolicy is enforceable in the service runtime environment. For each data assurance policy category introduced in Section 2, we have a unique state machine in a service provider's running environment to represent the persistent data object's lifecycle states and the involved state transitions. The actions for policy enforcement are encoded in the state machine and executed when state transition happens to the state machine.

The state machine-based policy enforcement approach works as follows:

1) For each persistent data object type and for each data assurance policy category, the system designer designs a state machine that represents the lifecycle states and transitions of the associated data object type. GEODAC provides template state machines for the introduced set of capabilities in the language, which can be tailored for individual service providers. In our approach, the granularity of a data object can be a file, a directory, a database record, a collection of database records, or an entire database table.
2) The system designer defines enforcement actions to either states or state transitions in the state machine.
3) The WS-DataAssurancePolicy for the persistent data object has configuration parameters listed for each policy assertion. These configuration parameters are mapped to the corresponding parameters in the state machine. This allows configurations of the state machine templates in the runtime for different service customers for their own data objects.
4) The enforcement component is built on top of a workflow engine. At runtime, when an event happens to the persistent data object, the state machine is activated and the event triggers the state machine to perform corresponding state transition. Thus the event drives the policy enforcement actions.

6.1 State Machine to Represent Data Lifecycle Management

Policies are attached to data at an arbitrarily fine granular level. A content management service could hold content for multiple organizations with each customer's data being separated into different file directories. If data is held in databases, individual records (or record collections) that belong to different customers may have different policies associated with them, especially if the database holds PII such as names and credit card numbers. Our framework assumes that each service provider interprets and executes the policies independently.

Policies are described as state machines that encode the lifecycle-related states for data, such as *created*, *updated*, *destroyed*, etc., and corresponding lifecycle events, such as *create-request*, *update-request*, *destroy-request*, for files, directories, database records, database tables, etc. Each state represents a different lifecycle stage of the data.

Actions are associated with either the states, or transitions between states. Actions associated with states are triggered at state entrance, while actions associated with state transitions are triggered when the state machine takes that transition out of a state.

Actions may consist of complex workflows or action sequences. They may involve external long lasting web service invocations. They may also involve human decision support such as content approval by multiple parties. Existing workflow technologies such as BPEL (BPEL4People) [41] or Windows Workflow Foundation (WWF) [33] can be used to describe action sequences.

Sometimes action sequences may themselves be better represented as state machines. A two-level state machine hierarchy can be constructed, where the parent state machine represents the lifecycle states, while the child state machines represent actions with their own specific action sequences. Consequently, a whole data assurance policy (to cover the whole lifecycle) can be described in a hierarchically composed workflow set, with a state machine at the top and the detailed action workflows at the leaves. The child state machine or the child sequential workflow is called a *child action workflow*.

Child action workflows are spawned from the parent workflow asynchronously with their own thread of control to support long delay times or human decision making. Events can be used to notify the parent workflow from the child workflow. Section 6.2.1 shows an example of simple parent-child event-based interaction.

Regarding the relationship to WS-DataAssurancePolicy defined by the service customer, as shown in Fig. 3, there is a state machine for each category of data assurance requirements. Each persistent data object type is associated to a set of state machine templates. Each state machine template enables enforcement of policy assertions of a specific category.

6.2 Data Assurance Policy Examples

To help understand our state machine-based approach to represent the enforceable data assurance policies, we provide two examples, the first one for data retention policy, and the second one for data migration policy.

6.2.1 Data Retention

Nullco sends its entire customer purchase history and customer profile information to the data mining analytics service provider to build the optimum set of customers to engage for the product marketing campaign. Nullco would like the data retention policy for both customer purchase history and profile information to be as follows:

(**R1**) The data will be destroyed at the end of the service contract. The exact time is specified at the time the service contract is signed (*DeleteBy* in Table 1);

(**R2)** After the data is deleted, an email notification is sent to the specified email address in Nullco (*EmailNotification* in Table 1).

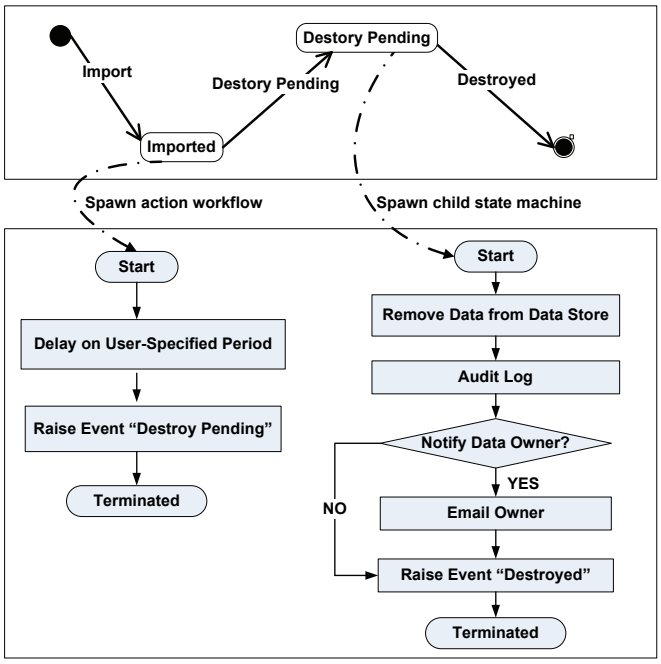


Fig. 8. State machine-based data retention policy.

Fig. 8 shows the complete state machine-based policy for data retention. When the data is *imported*, at the state initialization phase, a child action workflow is spawned, which starts the timer for a customer-defined time (specified in the WS-DataAssurancePolicy). At the end of the timer delay, an event *destroy-pending* is raised to the parent state machine, which will then transition to the state of *destroy-pending*, where a second child action workflow is launched to perform the actual data deletion action. Should the service customer choose to be notified for data deletion, an email notification will be sent, before the child action workflow terminates itself.

6.2.2 Data Migration

Suppose Nullco outsources its CRM database to a database service provider. The database contains PII such as customer names and addresses. As part of the marketing campaign, such customer data needs to be migrated to the Brochure Printing & Mailing service. A data migration policy can be attached to the entire database table for this purpose. The detailed migration policy consists of the following rules:

(**R1**) The candidate data receiver must be Safe Harbor [32] certified *(CertifySafeHarbor* in Table 1);

(**R2**) A data fingerprinting service [39] must be used to produce a unique copy of the database table for each receiver (*FingerprintData* in Table 1);

(**R3**) The data owner (Nullco) must be notified upon the completion of migration actions (*EmailNotification* in Table 1);

(**R4**) Once the contract is terminated, the data receiver must remove the data within two weeks and notify the data provider of data deletion (*DeleteBy* and *EmailNotification* in Table 1).

Fig. 9 shows the state machine which enforces rules R1, R2 and R3 at the data provider side. The top part describes general states of *created*, *imported*, and the final state (*destroyed*) associated with the data lifecycle. A child state machine is spawned at *imported* or *created* due to the state transition of *export*, in order to handle migration-related events and actions. The child state machine starts in state *data-migration-pending*, and initiates a workflow about migration procedure checking (shown in Fig. 10) to enforce R1 and R2. The workflow results in the event *migration-procedures-succeeded* or *migration-procedures-failed*. This results in the state moving to *data-eligible-to-migrate* or *data-not-eligible-to-migrate* state respectively. Similarly once the data has been transferred, a *data-transferred* event is raised to enable notification of the data-owner (R3).

Note that R4 is enforced at the data receiver (a different service provider to which the policy is propagated, as identified in Section 2.2.2) which supports a similar state machine policy to that shown in Fig. 9, when the transition occurs from state *imported* to the final state due to a *destroy* event.

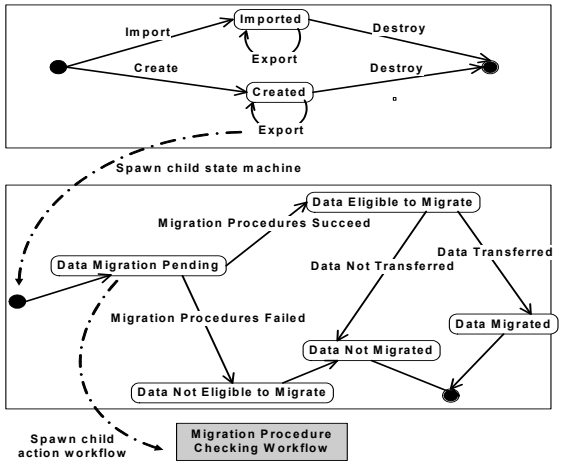


Fig. 9. State machine-based policy for data migration.

A child action workflow attached to the *data-migration-pending* state and spawned at the state initialization phase is shown in Fig. 10 to show the actions taken to determine whether the migration procedure succeeds or not and raises the corresponding event of *migration-procedures-succeeded* or *migration-procedures-failed*. The calls to the certification and fingerprinting services are used.

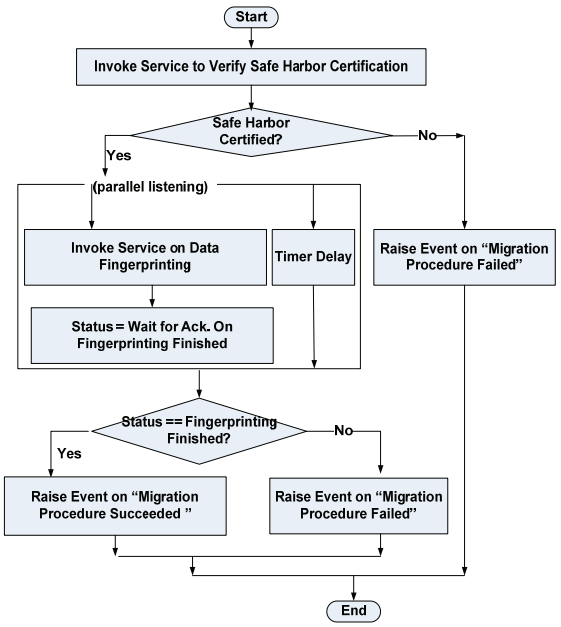


Fig. 10. Migration Procedure checking workflow.

6.3 Discussion on State Machine Feasibility

Section 6.2 shows two examples of the data assurance categories that we introduced in Section 2. In fact, we have expressed all the data assurance policy categories introduced in Table 1 except data confidentiality, using the presented machine-interpretable policy assurance framework. This is performed by defining required state machines and the detailed action sequences that need to be carried out for each state or transition in the state machines. The definition of actions depends on the granularity of data (record, table, database, file or directory). When needed, new state machines and associated actions can be defined for new capabilities in the framework.

6.3.1 Enforceable and Observable Actions

A general security control can be either a management control, an operational control, or a technical control [22]. Management and operational controls are difficult for a service customer to verify. In our approach, we rely on encoding actions in the state machine policy into the *enforceable actions* and the *observable actions* [28]. Enforceable actions reflect technical controls. Observable actions allow human actions which support management and operational controls to be documented clearly, tracked over time, and transparently have the observation results shared with service customers and/or auditors [36].

Because the enforceable actions are taken in the service environment, the service customer is not able to directly verify that the actions are actually performed correctly. Consequently, the service customer has to rely on the established trust relationship, and external auditors if needed. However, with observable actions, the service customers can still gain much more confidence from such transparent reporting on policy enforcement.

A control action of a policy assertion translates to an enforceable or observable action, or both. For example, with respect to data retention, the enforceable action is to remove the data when the retention time expires, and there are also two associated observable actions to: (i) perform audit logging on data deletion (such logs are later made accessible to service customers or auditors), and (ii) send email notifications.

6.3.2 Expression of Other Data Assurance Policy Categories

Data assurance policy categories, beyond those discussed in Section 6.2, can be expressed using this state machine-based approach. For each data assurance policy category, a state machine is needed to represent the data lifecycle and associated actions. In this section, we briefly illustrate how this can be done for some other data management categories that we identified in Section 2.

Data Appropriateness for Use. To ensure data appropriateness for Nullco's marketing campaign materials we expressed an approval process in the state machine-based policy framework to require approval of certain actions, such as the creation or modification of the marketing materials. The approvers can be humans, software services, or a combination of the two. This enables Nullco staff to review the marketing campaign materials before usage. To facilitate appropriateness checking, the state machine-based policy includes states such as *create-pending* and *update-pending*, and the corresponding events *create-approved* and *update-approved*. A *create-pending* event triggers an approval-related child action workflow, which issues requests to solicit approvals. Based on the received approval (disapproval) results, the child workflow will raise a *create-approved* (*create-disapproved*) event. The transition due to the *create-approved* event will have the action to accept the pending document, whereas the transition due to the *create-disapproved* event will have the action to discard the pending document.

Privacy Requirements. Regarding the control actions that we identified in Table 1, *RestrictLocation* and *Certify-SafeHarbor* can be enforced similarly by following what is described in Section 6.2.2 for data migration. With respect to *DataBreachNotification*, the state machine can accept the event of *DataBreached*, which is raised from an external business process that monitors the associated data (See Section 7.1 on how to deal with external business processes). When the *DataBreached* event is triggered, the predefined actions such as email notification to the data owners will happen.

Data Integrity and Data Availability. We can design the respective machine-interpretable policies completely based on observable actions. In contrast to other data assurance categories such as data retention and data migration, the policy execution for data integrity and data availability mainly involves external monitoring actions from the client-side, in order to build confidence in the server-side execution behavior based on the monitoring results. More specifically, to verify data integrity, we can design a state machine and the action workflows to query the outsourced database and perform verification on the query result, based on truth about the data that the owners know [42]. Similarly, for data availability, we can follow web site availability testing strategies, such as [16], to periodically generate artificial customer data, and per-

form data update or retrieval actions on the service. Repeated measurements of service availability build confidence in the future availability of the service.

7 Policy Enforcement: Architecture, Implementation and Evaluation

We have designed and implemented the GEODAC policy management and runtime enforcement framework for WS-DataAssurancePolicy.

7.1 Architecture

The high-level architecture to manage and enforce the WS-DataAssurancePolicy is shown in Fig. 11. Following the GEODAC framework, the customer-specific WS-DataAssurancePolicy ultimately is stored in the *WS-DataAssurancePolicy Repository.*

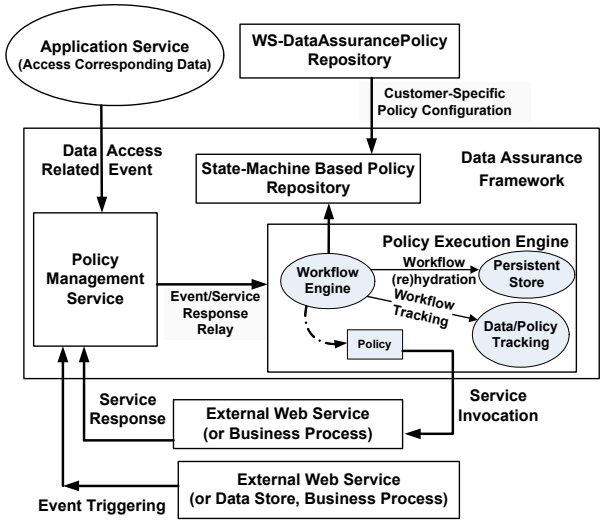


Fig. 11. Architecture for policy execution and management.

A *data access* event is raised from the application web service such as the content management service, typically from a service interception point defined in a service method. Each service interception point is installed with all event adapters that are responsible to generate the events for the involved data objects in the service method. The event's semantics are defined based on the corresponding state machine policy designed for the data object(s) involved in the service method call. Once a data access event, such as data creation, is sent to the *Policy Management Service* from the service interception point, the corresponding policy is triggered by the policy execution engine, resulting in an activated *policy execution instance*. Other events that relate to update, fetch, destroy, import, etc., of data are raised from the corresponding service interception points and drive state transitions within the activated policy execution instance, resulting in execution of the associated actions.

Event trigging at the service interception point is customer dependent. That is, if a customer opts out of the involved data object's data assurance policy, the event will not be triggered at the service interception point. Furthermore, the policy execution instance is also customer-specific. The policy that is actually being activated is based on the state machine policy template stored in the *State Machine-Based Policy Repository*, but it takes into account customer-specific WS-DataAssurancePolicy.

Similar to service interception points, external web services or business processes can also trigger events. The triggered event is encoded as a message delivered to the Policy Management Service and the service further routes the event to the corresponding policy execution instance.

During the policy execution, the policy execution instance can invoke operations on external web services or business processes such as to verify the Safe Harbor certification as shown in Fig. 10. Responses for asynchronous operations are sent back to the policy execution engine through the Policy Management Service.

The core of the policy execution engine is a workflow engine. To manage a large number of simultaneous state machines, the workflow engine supports hydration and de-hydration [33] such that the activated policy execution instance only resides in the runtime memory when a transition within it is trigged by an event. As mentioned earlier, events may be raised internally from child action workflows (see Section 6.1), or externally from an event or service response by external web services and business processes (shown in Fig. 11).

The Policy Management Service serves as the façade for the policy execution engine. To achieve scalability in a machine cluster, one policy execution engine runs on one machine and registers itself to the Policy Management Service. The Policy Management Service receives the messages from either service interception points or external web services/business processes, and deposits the message to the message queue that belongs to one of the active execution engines. Correspondingly, the chosen execution engine pulls the message from its message queue, and its message dispatcher translates the message into a local event in the workflow runtime. The event is then raised to the targeted policy execution instance. The involved workflow that corresponds to the policy execution instance may need to be hydrated into the runtime memory before the event is raised to the policy execution instance. In general, all state machines are persisted on shared storage and are hydrated when necessary. Therefore any node in the cluster can handle any event or message. However, in order to guarantee the total ordering of messages that are dispatched to the same targeted policy execution instance, the Policy Management Service's routing protocol needs to ensure that:

(i) All the messages that belong to the same policy execution instance will need to be routed to the same policy execution engine; and

(ii) If child action workflows are involved in the policy, we need to route the child workflow related message to the same policy execution engine that handles the policy execution instance. The correlation relationship between the child workflow and its parent is tracked by the *Data/Policy Tracking Service* shown in Fig. 11.

To satisfy the above requirements, our routing protocol uses a deterministic hashing function over the topmost parent workflow identifier (which represents the

unique identifier of the policy execution instance), with the hashing result being an integer number that corresponds to a particular machine in the cluster.

Each state machine policy and its corresponding policy execution instances are designed for a data assurance category (e.g., data retention or data migration). As long as the event source, which is either a service interception point or an external web service/business process, can clearly identify the triple {data object's unique identifier, customer identifier, policy category}, the event will be raised to the correct state machine policy execution instance. Consequently, our WS-DataAssurancePolicy enforcement framework supports customer-specific policies that can involve multiple data assurance categories (e.g., both data retention and data migration) for a data object.

7.2 Implementation

We have implemented the prototype of the WS-DataAssurancePolicy enforcement framework described in Section 7.1 in a Microsoft .NET web service environment. The service modules, i.e., *WS-DataAssurancePolicy Repository,* Policy Management Service, and *Data/Policy Tracking Service*, are all implemented as web services that can be distributed into different machines in a machine cluster that forms the policy enforcement platform. The message queue is implemented as a .NET remoting object server on each machine from which the local policy execution engine pulls the messages. With the web service interface exposed for each message queue on each machine, the Policy Management Service can push the messages to the selected queues in the machine cluster.

Our state machine policy is encoded in Microsoft Windows Workflow Foundation (WWF) [16], with a top-level state machine being described as a WWF *state machine workflow*. If a child action workflow represents an action sequence, it is described by a WWF *sequential workflow*. If it is a state machine (like the one shown in Fig. 9), it is described by a WWF state machine workflow.

In a policy execution instance, the way for the parent workflow to invoke the child workflow is through the WFF workflow activity pattern called *InvokeWorkflow*. We use the communication mechanism in WWF called Local Communication Services (LCS), to have the child workflow internally (within a workflow runtime) raise an event to the parent workflow. Raising the event from the child workflow to its parent requires that the child know the parent's instance identifier. We solved this by *data binding* supported by the *InvokeWorkflow* activity pattern. More specifically, data binding happens to the data defined as *Dependency Property*. The parent workflow's Dependency Property *MyWorkflowId* is bound to the child workflow's Dependency Property *ParentWorkflowId.* Once the child workflow is activated, the value of the data is transferred accordingly.

All the policy configuration related parameters defined in WS-DataAssurancePolicy have their counterparts defined as Dependency Properties in the topmost workflow. The configuration parameters and the Dependency Properties need to be one-to-one mapped. The mapping happens when the policy execution engine creates the topmost workflow instance of the policy. More specifically, the configuration parameters are passed as the workflow instance creation's input arguments. Once the workflow is started, the configuration parameters are mapped to its Dependency Properties. If the configuration parameters are actually used in the child action workflows, then the child action workflows need to have the corresponding Dependency Properties defined and the customer-specific WS-DataAssurancePolicy parameters can then be passed down through the same data binding process.

Finally, policy execution requires tracking of the relationships between child action workflows and their parents. Besides the parent/child workflow relationship tracking, our framework also supports correlation identifiers [41, 33] to track application-level relationships. For example, we have a tracking identifier for event *Data Export* (described in Section 6.2.2) and subsequently for event *Data Transferred* (also described in Section 6.2.2). If such correlation (tracking) identifier is required, the corresponding workflow and its child action workflow need to both have the corresponding Dependency Property *CorrelationId*, and allow the *CorrelationId* to be bound from the parent to the child workflow. Both parent/child relationships and correlation identifiers are tracked in the Data/Policy Tracking Service, by having the child workflow report such relationships to Data/Policy Tracking Service at the initial state's entrance phase.

7.3 Enforcing Example Policies in GEODAC

We have realized data retention and data migration, the two policies examined in Section 6.1, in WWF. We have integrated them with the WS-DataAssurancePolicy enforcement framework. In our experimental setup, a machine in the cluster is devoted to host all the web services shown in Fig. 11. A separate group of machines (two in our experiments) are devoted to host policy execution engines. A client application (on a different machine) invokes the application service, *DataMiningService*, which has service interception points installed with event adapters. By invoking a sequence of the *DataMiningService*'s methods, such as *CustomerDataStoreInitialization*, *CustomerDataImport*, *CustomerDataExport*, etc., the client application triggers the corresponding data retention or data migration policy in one of the policy execution engines. All the .NET DLLs that represent the runtime version of the state-machine policies are distributed to each engine. The workflow instance identifier is created randomly at the interception point, and gets assigned to the activated workflow instance later. Thus the Policy Management Service's routing protocol is able to deterministically choose a policy execution engine for policy execution. The prototype demonstrates the feasibility of enforcing customer-specific data assurance policies using the GEODAC framework.

8 Related Work

There is much work proposing languages and support to express and enforce message-level security policies in Web services (see [4][8][27][43] for surveys). We discuss

related work in two categories: data assurance policy specifications, and data assurance policy enforcement.

Data assurance policy specifications. There has been a lot of work on defining languages for the specification of Web services security policies [4][8][1][21]. However, most of these efforts have been focused on message level security requirements such as confidentiality, integrity, authentication and trust [20], or resource protection mechanisms for access control [6][7]. Standardized policy specification languages such as WS-SecurityPolicy [25] and XACML [23] fall into the same category.

P3P is a language for specification of personal privacy policy for Web sites [40]. The set of privacy concerns in our work is focused on enterprise needs and in the context of outsourcing. The work of [12] also presents abstractions for expressing privacy concerns at the business protocol level. Our framework is not limited to expressing only privacy concerns but all related data assurance policies. In addition, our work introduces the enforcement mechanisms for the policies.

Some recent work presents high level frameworks for representing security requirements (e.g., [27]) and at the business process level (e.g., [15][20]). However, these works also focus on message-level security polices and do not provide mechanisms for enforcing policies in service environments. We have built on top of the WS-Policy framework [2] to create WS-DataAssurancePolicy, which is an extensible language for data assurance policies.

Data assurance policy enforcement. In the service oriented environment, there exist two policy enforcement points, one at the service messaging level and the other at the backend processing level (e.g., access control). The message-level enforcement has been well investigated by the WS-Security related techniques [25][43]. In this paper, our data assurance policy and its enforcement are focused on the persistent data objects that are stored, processed and migrated by the outsourced service environments.

Access control is a particular type of enforcement mechanism in backend processing. There are works in this area (e.g., [7]) that focus on access control mechanisms in outsourced services environments, which are complementary to our work in this paper.

Another complementary theme of work is distributed usage control [28]. This work discusses the handling of sensitive data passed from the data provider to its consumer by defining obligations that can travel with the data. In our data assurance framework, detailed obligations are expressed as the child action workflows attached to specific states and transitions (that describe the situation). Our solution provides a practical approach to enforce obligation related policies in outsourced service environments. Furthermore, our framework supports policies other than usage control such as data appropriateness for use, data availability, data migration, etc.

State machines have long been used to describe and enforce expected system behaviors. Trust-Serv uses state machines to express trust negotiation policy models [35]. However, it focuses on external interactions to establish trust. PRIME [29] supports privacy obligation management, with the focus on expressing and enforcing data retention policies. A policy engine is presented in [5] to handle obligation-related policies through dynamic policy construction from policy templates. In contrast, our data policy assurance framework offers much broader data assurance aspects and more flexible policy customization capabilities to service customers.

Our work is complementary to document management systems that support workflows, such as Microsoft SharePoint. Indeed, our solution addresses a broad range of data resources beyond documents and can be applied in much more complex cross-organizational situations.

Our work provides application-level data assurance capabilities beyond data access control, which is currently in cloud computing platforms. Hardware-platform-as-a-service (e.g., Amazon.com) focuses on hardware virtualization and is not aware of high-level business data requirements. Therefore, data encryption is the only recommended mechanism to protect business data in persistent stores such Amazon.com's Simple Storage System (S3) [1], which as we have shown is not sufficient for a broader range of data assurance categories identified in this paper. Software/platform-as-a-service providers for application development and hosting (e.g., Salesforce.com) currently provide only role-based access control for data assurance [31].

9 Conclusion and Future Work

In this paper, we have presented a framework and methodology for identifying, capturing and enforcing data assurance policies. To the best of our knowledge, this is the first work which characterizes the data assurance policy requirements in outsourced services environments, and proposes a mechanism for enforcing customer-specific policies on the persistent data stored in service provider environments.

We have proposed the GEODAC framework to address some of the data assurance issues which arise when using outsourced services. GEODAC includes the extensible WS-DataAssurancePolicy language which allows service providers to specify the data assurance capabilities they can provide for received data, and service customers to configure these capabilities for their unique situation. We presented a data assurance console which provides an intuitive interface for users to specify their needs without needing to create policies directly in the WS-DataAssurancePolicy language. We implemented a prototype enforcement mechanism for some of these policies based upon state machines which allows binding the policies to data items at various levels of granularity. In cases where the service customer cannot directly verify the enforcement actions in the service provider environments, the framework enables indirect observations to provide increased confidence of policy compliance.

The GEODAC framework also opens new avenues for investigating the more general area of distributed policy enforcement. In particular, the framework is designed to

enable the propagation of data assurance policies along with data when data migrates from a service provider to sub-contractors, and subsequent enforcement of the policies using the framework. We plan to use the framework in a multi-party service scenario to further explore its feasibility and refine it. Another next step is to evaluate the scalability of the framework in a larger context (100 machines).

REFERENCES

[1] Amazon Web Services, "Amazon Web Services: Overview of Security Processes," Sept. 2008, accessible at http://s3.amazonaws.com/aws_blog/AWS_Security_Whitepaper_2008_09.pdf.

[2] S Bajaj and et al. Web Services Policy 1.2 - Framework (WS-Policy), www.w3.org/Submission/WS-Policy/, April 2006.

[3] S. Bajaj, Web Services Policy Attachment (WS-PolicyAttachment), http://specs.xmlsoap.org/ws/2004/09/policy/ws-policyattachment0904.pdf.

[4] P. A. Bonatti, et al. Rule-based policy specification: State of the art and future work. Technical report, Working Group I2, EU NoE REWERSE, August 2004. http://rewerse.net/deliverables/i2-d1.pdf.

[5] M. Casassa Mont, F. Beato, "On Parametric Obligation Policies: Enabling Privacy-aware Information Lifecycle Management in Enterprises," 8th IEEE International Workshop on Policies for Distributed Systems and Networks, 2007.

[6] E. Damiani, et al: Selective Data Encryption in Outsourced Dynamic Environments. Electr. Notes Theor. Comput. Sci. 168: 127-142 (2007).

[7] E. Damiani, et al.: Metadata Management in Outsourced Encrypted Databases. Secure Data Management 2005: 16-32.

[8] N. Damianou et. al., A Survey of Policy Specification Approaches, 2002, http://www.doc.ic.ac.uk/~mss/Papers/PolicySurvey.pdf.

[9] Directive 95/46/EC of the European Parliament and of the Council, http://www.cdt.org/privacy/eudirective/EU_Directive_.html.

[10] M. Fratto, "Internet Evolution – Cloud Control," Information Week, Jan. 2009. Accessible at http://www.internetevolution.com/document.asp?doc_id=170782&print=yes.

[11] C. Gutiérrez, Eduardo Fernández-Medina, Mario Piattini, "A Survey of Web Services Security," ICCSA (1) 2004.

[12] R. Hamadi et. al, Conceptual Modeling of Privacy-Aware Web Service Protocols, 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), 11-15 June 2007, Norway.

[13] Hibernate, http://www.hibernate.org/

[14] HIPPA, http://www.hhs.gov/ocr/privacy/index.html

[15] M. Jensen, S. Feja, "A Security Modeling Approach for Web-Service-Based Business Processes," 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2009.

[16] Keynote Systems, http://www.keynote.com/.

[17] P. Krishna and K. Karlapalem, "Electronic Contracts," IEEE Internet Computing, Volume 12, Issue 4, July-Aug. 2008 Page(s):60 – 68.

[18] J. Li et. al, "A Policy Framework for Data Management in Services," Proc. of 4th International Workshop on Dependability Aspects on Data Warehousing and Mining applications, Mar. 2009.

[19] Robert W. Lucky, Cloud Computing, IEEE Spectrum, Volume 46, Issue: 5, Page 27, May 2009.

[20] M. Menzel et. al, Security Requirements Specification in Service-oriented Business Process Management, 2009 International Conference on Availability, Reliability and Security, March 2009.

[21] H. R. Motahari Nezhad, H. Skogsrud, B. Benatallah, F. Casati, Securing Service-based Interactions: Issues and Directions. IEEE Distributed Systems Online (Web Systems Topic), Arpil, 2005.

[22] NIST, "Recommended Security Controls for Federal Information Systems and Organizations," NIST Special Publication 800-53 Revision 3.

[23] OASIS, eXtensible Access Control Markup Language (XACML), Version 1.0, Feb. 2003.

[24] OASIS, WS-Security, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[25] OASIS, WS-SecurityPolicy 1.2, July 2007, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html

[26] PCI DSS (Version 1.2), Oct. 2008. https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html.

[27] T. Phan et. al., Quality-Driven Business Policy Specification and Refinement for Service-Oriented Systems. In proc. ICSOC 2008.

[28] A. Pretschner, M. Hilty, and D. Basin, "Distributed Usage Control," Communications of the ACM, Sept. 2006/Vol. 49, No. 9, pp. 39-44.

[29] PRIME Project, https://www.prime-project.eu/.

[30] Rack Space, "Intrusion Detection Systems (IDS)," http://www.rackspace.com/downloads/pdfs/IDSOverview.pdf

[31] C. Roth, D. Carroll, and N. Tran, Creating On-Demand Applications: An Introduction to the Apex Platform, Book accessible at http://developer.force.com.

[32] Safe Harbor Home Page, US Department of Commerce, http://www.export.gov/safeharbor/index.asp.

[33] D. Shukla, B. Schmidt, *Essential Windows Workflow Foundation*, Addison Wesley, 2006.

[34] D. Sims, TMCnet, "Gartner Finds SaaS Market to Hit $19.3 Billion by 2011", http://www.tmcnet.com/usubmit/2007/03/07/2398768.htm.

[35] H. Skogsrud et. a., "Trust-Serv: Model-Driven Lifecycle Management of Trust Negotiation Policies for Web Services," Proceedings of the 13th Intl. conference on World Wide Web, pp. 53-61, 2004.

[36] Statement on Auditing Standards No. 70 (SAS 70), American Institute of Certified Public Accountants (AICPA) http://www.aicpa.org/download/members/div/auditstd/AU-00324.PDF.

[37] The Breach Blog, "BNY Mellon Shareowner Services loses backup tape," http://breachblog.com/2008/03/27/bny.aspx

[38] United States National Information Assurance Glossary, http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf.

[39] E. Uzun and B. Stephenson, "Security of Relational Databases in Business Outsourcing," HP Labs Report HPL-2008-168.

[40] W3C, Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P/.

[41] WS-BPEL 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[42] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity auditing of outsourced data," Proceeding of VLDB'07, pp. 782-793.

[43] A. Singhal, T. Winograd, K. Scarfone, Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-95, August 2007. http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf.