GEODAC: A Data Assurance Policy Specification and Enforcement Framework for Outsourced Services

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

HP Laboratories
HPL-2009-357 (R.1)

Keyword(s):
Security and Privacy in Services; Security and Privacy Management in Data Collection, Transformation and Dissemination; Service Oriented Computing; Software as a Service; Services Delivery Platform and Methodology

Abstract:
Many cloud service providers offer outsourcing capabilities to businesses using the software-as-a-service delivery model. In this model business critical data needs 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. There exist challenges to express service data assurance capabilities, capture customers' requirements and 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 data assurance policies in service providers' environments. High-level policy statements are backed in the service environment with a state machine-based representation of policies in which each state represents a data lifecycle stage. Data assurance policies that define requirements on data retention, data migration, data appropriateness for use, etc., can be described and enforced. The approach has been implemented in a prototype tool and evaluated in a services environment.

External Posting Date: April 30, 2010 [Fulltext] - Approved for External Publication

Internal Posting Date: April 30, 2010 [Fulltext]

To be published in IEEE Transactions on Services Computing, Accepted in February 2010.

GEODAC: A Data Assurance Policy Specification and Enforcement Framework for Outsourced Services

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

Abstract—Many cloud service providers offer outsourcing capabilities to businesses using the software-as-a-service delivery model. In this model business critical data needs 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. There exist challenges to express service data assurance capabilities, capture customers' requirements and 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 data assurance policies in service providers' environments. High-level policy statements are backed in the service environment with a state machine-based representation of policies in which each state represents a data lifecycle stage. Data assurance policies that define requirements on data retention, data migration, data appropriateness for use, etc., can be described and enforced. The approach has been implemented in a prototype tool and evaluated in a services environment.

Index Terms—Security and Privacy in Services; Security and Privacy Management in Data Collection, Transformation and Dissemination; Service Oriented Computing; Software as a Service; 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 how 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) is the latest software delivery mechanism [40]. Many businesses are eager to take advantage of these advances to run business functions in a more cost-effective manner by outsourcing them to cloud service providers.

While this delivery model provides many opportunities, there are 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 [22]. This problem is related to the issue of *information assurance*, which according to the U.S. National Information Assurance Glossary [45], 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 policy" to describe the set of information assurance requirements and enforcement capabilities to manage data in compliance with regulatory and corporate policies. Examples of regulatory compliance include the Data Protection Directive under EU privacy law [10], HIPAA [16] and PCI DSS [31].

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 is required. While many people have recognized the urgent need for data assurance mechanisms [11], these aspects have not received adequate attention. Existing policy languages are insufficient to express such requirements.

2- Businesses are reluctant to share business critical data like customer lists, sales data, or financial data. While data handling policies can be specified in service level agreements or contracts [19], these contracts are usually written in legal terms and do not contain fine-grained enforceable policies to manage data during operations. Even when the service provider agrees to these fine-grained policies, it is possible for data to be mishandled inadvertently because multiple service providers are often involved in delivering any service [44]. Thus it is important to develop policy assurance frameworks which handle customer-specific requirements to provide data assurance in service provider environments.

3- The selected data assurance policies have to be realized in the environments of the service providers. Currently, there is no systematic support or mechanism to enforce the customer's data assurance policies for persistent data objects (e.g., customer list, credit card number, etc.) used and stored within the environments of service providers. Existing approaches use network-level techniques, which are not adequate [1, 35].

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 environments. 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 a data assurance policy language, its grammar and an XML-based representation called WS-DataAssurancePolicy, which allows service providers to express data assurance 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 consisting of a methodology for service providers and customers to use the proposed data assurance policy language to enforce customer-specific data assurance requirements in service providers' environments. It enables security experts and system designers from service customers to work together to define data assurance policies. It automatically translates the customer-specific requirements into WS-DataAssurancePolicy policies.

4- We propose a runtime enforcement mechanism for service providers to enforce policies defined in WS-DataAssurancePolicy. Policy enforcement models are expressed in state machines with states representing lifecycle stages of the data. The state machine-based enforcement model incorporates both *enforceable* and *observable* control actions on data. Our model allows control actions to be long-lasting and/or human-dependent.

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

It should be noted that the current paper significantly extends our earlier work on enforcing data assurance policies in service providers' environments [21]. In particular, the novel contributions of this paper include items 1 (characterization of requirements), 2 (WS-DataAssurancePolicy) and 3 (GEODAC framework for policy customization and binding) above, and also extending the runtime environment 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 GEODAC data assurance policy language and its XML representation called *WS-DataAssurancePolicy* language. In Section 5 we describe a data assurance console for customers to configure data assurance policies without needing to author policies directly in WS-DataAssurancePolicy. In Section 6 we describe the state machine-based policy enforcement model and mechanisms. Section 7 presents the policy enforcement platform architecture, its implementation and experiments. In Section 8 we discuss related work, and we conclude the paper 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 as an illustrative example throughout the paper.

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.

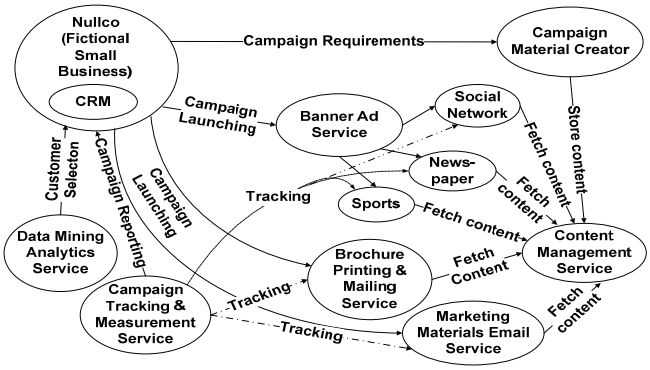

Figure 1. Services composed for on-line marketing.

Shown in Figure 1, the marketing campaign relies on services from ten different providers. A data mining analytics service identifies campaign targets, 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 customer contact information. The banner ad service subcontracts its work to other 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 campaign effectiveness.

2.2 Requirements for Data Assurance

Data assurance requirements must be considered in outsourced services engagement. In this section, we list the important data assurance requirements, and define necessary control actions for each requirement. A *control action* specifies concrete activity 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., [7][8][41]), but it is not discussed here.

We designed the GEODAC framework to enable specification and enforcement of all the controls necessary to safely outsource critical business functions, regardless of the level of abstraction of the controls. Table 1 summarizes the data assurance requirements and associated con-

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

Requirement	Control Action(s)	Control Label(s) in Policy (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*
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 or subcontractors	*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 retrieval and transmission	*CheckIntegrity*
Usage Appropriateness Requirements	Enable approvals of actions and spot-checks	*RequireApproval*

trol actions we identified, which are described next.

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 [38]. 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 [10] requirements. A control action is needed to restrict selection of services to those which certify Safe Harbor compliance.

Data breach notification: Many countries and states in the USA have data breach notification requirements for PII. To meet its notification requirements under these laws, a notification action is needed to ensure that Nullco is informed of suspected breaches of personal data.

2.2.2 Data Migration Requirements

The outsourcer may migrate data to subcontractors, which can be a source of data breaches [44]. This does not absolve the data owner of legal responsibility for the data. Four control actions are identified to address this risk.

Pre-migration checks: Nullco may have various requirements which govern the selection of services in addition to the Safe Harbor verification discussed above. Reputation services may be consulted to aid in selection. Nullco may even wish to pre-approve each migration of data. In general, a variety of pre-migration checks may need to be performed to ensure that data migration to other services is in compliance with policy.

Data fingerprinting: To reduce the likelihood of data theft and enable tracing of data leaks, Nullco may fingerprint a data set such as its customer list before sending it to service providers [46]. By providing each service with a uniquely marked data copy, it is possible to trace a data leak to the service which received that unique copy.

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.

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 wishes to place limits on how long data can be retained by the various services 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 Requirements

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. In addition to necessary 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 and prevent data leaks [44].

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 to preserve data confidentiality.

2.2.5 Data Availability Requirements

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. Thus, our framework does not directly enforce this requirement, but provides monitoring on availability (see Section 6.3).

2.2.6 Data Integrity Requirements

If data cannot tolerate errors, integrity checks are required when interacting with the service and within that service to quickly detect data corruption. The integrity checking related control action can adopt a hash function to verify proper data transmission and storage.

2.2.7 Appropriateness for Use Requirements

When a large percentage of a corporation's data is generated and maintained by outsourcers, it becomes 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 use to ensure they meet Nullco's brand standards. Often spot-checks of data quality by either software or humans are needed.

2.3 Enforceable and Observable Control Actions

A general security control can be either a management control (e.g., certification, security assessments, etc.), an operational control (e.g., contingency planning, incident response, etc.), or a technical control (e.g., access control, system and communications protection, etc.) [25]. Management and operational controls are difficult for a service customer to verify.

In our work, we adopt two types of control actions, *enforceable actions* and *observable actions* [33]. Enforceable actions reflect technical control actions. Observable actions allow management and operational controls to be documented clearly, tracked over time, and have results transparently shared with customers and auditors [43]. An example of observable actions is audit logging that allows each policy decision and enforcement step to be observed and ensures that the policy has been enforced in an irrefutably provable manner. This can be facilitated with mechanisms such as TPM and signature [4].

A control action is either enforceable or observable, or both. For example, for the data retention requirement, the enforceable action is to remove the data when retention time expires. The requirement can also have two observable actions: (i) perform audit logging on data deletion and (ii) send email notifications. Auditing is a generic control action to all data assurance requirements and thus does not show up in Table 1. Notification through *Email-Notification* is identified in Table 1.

The enforceable actions in the service environment may not be directly verifiable by the customers. However, with observable actions that produce accessible artifacts to service customers, the service customers can gain more confidence from transparent reporting on policy enforcement. The service customer may also rely on the established trust relationship and external auditors [43].

We believe that trust relationships could be established through similar approaches to business relationships that are formed in society such as business engagement, reputation [36], and business contracts. Nevertheless, handling this issue is out of the scope of this paper.

3 The GEODAC Framework: Overview

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

Figure 2 shows the overall GEODAC framework in a schematic way. The service provider publishes its data assurance capabilities for each service on individual data object types that are involved in the service. The capabilities are described as parametric statements with acceptable values for each configurable parameter. For example, the marketing service may offer a data retention capability for marketing collateral. Thus, the marketing collateral data type will have a policy statement with a parameter for when to destroy the data.

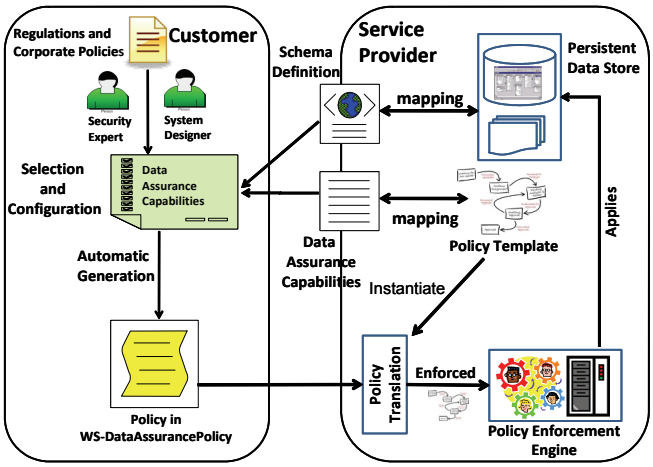

Figure 2. The GEODAC framework for data assurance policy specification and enforcement.

A compliance or security expert of the customer 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 for each data type and configure them in accordance to regulations and corporate policies.

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. Two aspects of data assurance policies are 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 parameter value defined for each data object type.

During the process of policy configuration, 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 WS-DataAssurancePolicy (described in Section

4). The policy 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 policy, binds the published data types into concrete persistent data object types, and instantiates state machine-based enforcement model templates. The enforcement engine uses the mapping of published data schema into persistent store schema to enforce policy for different data objects at different levels of data granularity, e.g., database, table, record, file directory and file.

4 Data Assurance Policy Specification Language

The data assurance policy language allows service providers to present their data assurance capabilities, and service customers to express their data assurance requirements. A service customer chooses a service and configures the service capability specification according to its data assurance requirements. We present the grammar of the language as well as its XML-based representation by extending the WS-Policy language [3] to accommodate data assurance policies. The XML-based policy language is called the *WS-DataAssurancePolicy* language.

4.1 Data Assurance Policy Specification

Data assurance policy specification aims to specify both the structural relationships among assurance capabilities, services and data, and the configuration of the defined assurance capabilities. The design of our policy language is guided by the data assurance requirements identified in Section 2.

Our language focuses on *persistent data* that is stored and processed in the service environment and potentially further transferred to subcontractors' service environments. The data may have different representation while in transit over the communication channel (referred to as *transient data*) or when it is stored. A persistent data specification has two attributes, the *persistent reference* that specifies the URL where the data type definition within the service environment is stored, and the *transient reference* that specifies the URL where the transient data representation format is stored. In practice, a transient reference can point to the data definition within the service's public WSDL document.

The mapping of published data types identified through transient references, to persistent data objects identified through persistent references, may be cumbersomely complex. Therefore, in our data assurance framework, a transient data object is restricted to have one-to-one mapping to a persistent data object in the service environment. Such mapping restriction actually has been adopted by high-level data mapping frameworks such as Hibernate [15] or high-level business objects programming environments such as Salesforce.com [37].

At the top level, our policy specification language contains the *service capability specification* that describes data assurance capabilities offered by the service provider, and the customer's *assurance policy specification* that is a strict subset of the service capability specification.

4.1.1 Service Capability Specification

There are three key concepts in the definition of service capabilities: an *assurance capability* represents a data assurance capability supported in the service environment (example capabilities are listed in the first column of Table 1), a *control action* represents a mechanism implemented in the service environment to enforce an assurance capability (example control actions are listed in the second and third column of Table 1), and the *service capability* that specifies the supported assurance capabilities on persistent data stored in the service. One or more control actions are needed to realize an assurance capability.

A control action may have attributes. Each attribute is configurable by a service customer. For example, the control action that supports data retention, *DeleteBy*, has an attribute *NumberOfRetentionDays* to specify the number of days after which the data object will be deleted, from the date that the data arrives and is stored in the service environment. Attributes can be absent if no configurable attributes are involved. A control action can optionally include constraints over the attributes specified in the control action. For example, in the control action *DeleteBy*, the constraint could be that *NumberOfRetentionDays*<3650, to denote that data retention cannot exceed 10 years. We adopt Object Constraint Language (OCL) to specify attribute constraints [29].

For some control actions, the service provider may only allow selection of one action among a set for fulfilling a given capability. These control actions are called *mutually exclusive*. For example, the two controls, *DeleteAtEndOfEngagement* and *DeleteBy*, can be considered mutually exclusive by the service provider, due to the service's specific implementation. The service customer is allowed to pick only one of these two control actions.

Once all the controls and assurance capabilities are defined, the service capability specification presents the assurance capabilities on persistent data types within a service. We define service capability specification as follows.

Definition 1 (Service Capability Specification). The capability specification CS of a service provider includes a set of persistent data types *D*. To each data type $d \in D$ a set of capabilities C and attributes U is associated where U is a set of attributes on the data type. Each capability $c \in C$ represented is associated with a set of control actions *A* nominated to realize c in the service environment. Each control action $a \in A$ comes with a set of configurable attributes Z and a set of constraints L defined on attributes in Z. For each $z \in Z$, we have the tuple $< z, V_z >$ where V_z is the set of acceptable values for z.

```
<specifications>            ::= <services capabilities> | <services specifications>
<services capabilities>     ::= namespace identifier
                                '{' [<control action>]+ [<assurance capability>]+ [<service capability>]+ '}'
<control action>                ::= define control identifier '{' [<attribute >]* [<constraint>]* '}'
<assurance capability>      ::= define capability identifier '{' [exclusive '{' [control identifier]+ '}']* [control identifier]+ '}'
<service capability>        ::= service identifier provides '{' [<attribute definition>]+ [data capability]+] '}'
<data capability>                   ::= data identifier '{' [<attribute definition>]+ ' }' holds '{' [capability identifier ]+ '}'
<services specifications>           ::= namespace identifier '{' [<service specifications>]+ '}'
<service specifications>            ::= service identifier required '{' [<attribute assignment>]* [<configured capability>] + '}'
<configured capability>             ::= data identifier holds '{' [<configured assurance capability>]+ '}'
<configured assurance capability>  ::= capability identifier '{' [<configured control action>]+ '}'
<configured control action>     ::= control identifier '{' [<attribute assignment>]* '}'
<attribute definition>      ::= attribute identifier ':' <attribute type> | attribute identifier ':' <attribute type> '=' <default value>
<attribute assignment>  ::= attribute identifier ':' <attribute type> '=' <assigned value>
```

Figure 3: Definition of GEODAC Data Assurance Policy Language

The GEODAC language's grammar is shown in Figure 3. Figure 4 shows a simple example for expressing *data-retention-capability* capability for service *CustomerDataMiningService* following the defined grammar. The entire specification is under the namespace of *SuperAximServices*. We define two control actions, *DeleteBy* and *DeletAtEndOfEngagement* for this capability. The control *DeleteBy* has a configurable attribute called *NumberOfRetentionDays*. We associate this service capability to *CustomerDataMiningService*. It includes a service level attribute called *service_definition*. This service supports a persistent data type called *CustomerPurchaseHistory*. This persistent data type has its transient data reference and persistent data reference specified. This persistent data type supports the capability *data-retention-capability*.

```
namespace SuperAximServices {
   define control DeleteBy {
       attribute NumberOfRetentionDays: integer;
       constraint NumberOfRetentionDays < 3650;
   }
   define control DeleteAtEndOfEngagement {   }
   define capability data-retention-capability {
       exclusive {
              control DeleteBy;
              control DeleteAtEndOfEngagement;
       }
   }
   service CustomerDataMiningService provides {
      attribute service_definition:URL=
          http://www.example.com/DataMiningService?wsdl;

      data CustomerPurchaseHistory {
        attribute transientref:URL =
            "http://www.example.com/DataMining.wsdl
                ?object=CustomerPurchaseHistory";
        attribute persistentref:URL=
            "http://www.example.com/DataMining/
                schema?data=CustomerPurchaseHistory";
        holds {
            capability data-retention-capability;
            …. //other capability
        }
      }
   }
}
```

Figure 4: Service capability specification example

4.1.2 Data Assurance Policy Specification

The service customer considers data assurance capabilities from the service provider and configures them based on the customer's data assurance requirements. The configuration is performed for configurable attributes of control actions by selecting an appropriate value, subject to the constraints in the control action.

A configured assurance capability specification forms the policy specification for the service customer, which is called *data assurance policy specification* defined as follows:

Definition 2 (Data Assurance Policy Specification). A policy *P* is a configured service capability specification *CS* for a customer in which for control action $a \in A$ of a capability *c* of data type $d \in D$, the set of attributes Z conforms to constraints L. For each $z \in Z$ in *P*, we have $< z, v >$ and $v \in V_z$ meaning that a specific value is chosen for z. In this paper, the term data assurance requirement is interchangeable with data assurance policy specification.

```
namespace NullCoSelectedServices {
  service SuperAximServices::CustomerDataMiningService required {
      data CustomerPurchaseHistory holds {
          capability data-retention-capability {
              control DeleteBy {
                attribute NumberOfRetentionDays: integer= 30 ; /*days*/
              }
          }
          .... //other capability configuration
      }
      …. //other persistent data associated configuration
  }
}
```

Figure 5: Data assurance policy specification example

Figure 5 shows a data assurance policy example based on the service capability specification presented in Figure 4. The policy is defined under the namespace of *NullCoSelectedServices*. The policy is the result of configuration of the capabilities defined for service *CustomerDataMiningService*. The persistent data type involved in this service is *CustomerPurchaseHistory*, which holds the capability *data-retention-capability*. The configuration happens to the control action *DeleteBy*, with its attribute *NumberofRetentionDays* being assigned the value of '30' days.

Table 2: Data assurance vocabulary following assurance requirements identified in Table 1.

Capability	Control Actions	Control Action Related Attributes
Privacy Protection	RestrictLocation	AllowedRegions (enum)
	CertifySafeHarbor	SelectedService (URL)
	DataBreachNotification	NotificationEmailAddress (string) TimeToNotification(enum)
Data Migration	ReputationCheck	ConsultedService (URL)
	FingerPrintData	SelectedCreditCardCompany (URL) ProbabilityMissRate (float)
	EmailNotification	EmailReceiver(string) TimeToNotification (enum)
	PropagatePolicies	MethodToPropgatePolicies (enum) ReceiverEmailAddress (string) AllowedTimeDelayToPropagatePolicies(date)
Data Retention	DeleteBy	NumberOfRetentionDays (integer)
	DeleteAfterLastUpdate	NumberOfRetentionDays (integer)
	DeleteAtEndOfEngagement	(no configurable attributes)
	EmailNotification	EmailReceiver(string) TimeToNotification (enum)
Data Confidentiality	EncryptStorage	EncryptionKeyType (enum) EncryptionKeyLength(enum) OnlineKeyBackup(Boolean) NumberOfKeyBackupCenters (integer)
	EncryptTransmission	EncryptionKeyType (enum) EncryptionKeyLength(enum)
Data Integrity	HashDataContentOnStore	HashAlgorithmChosen(enum)
	VerifyDataIntegrityOnRetrieval	(no configurable attributes)
Data Usage Control	RequireApprovalOfActions	Actions(enum), Approvers(string)
	ObligationControl	RequiredAction (enum)
		AllowedTimeDelay(enum)

4.1.3 Data Assurance Vocabulary

We define in Table 2 the data assurance vocabulary corresponding to the data assurance requirements identified in Table 1. This vocabulary consists of the data assurance capabilities (the first column of Table 2), the control actions associated to each capability (the second column of Table 2) and the configurable attributes involved in each control action (the third column of Table 2). For example, with respect to the Privacy Protection capability, we have identified three control actions: *RestrictLocation*, *CertifySafeHarbor* and *DataBreachNotification*. *RestrictLocation* has an attribute called *AllowedRegions*, which is an enumerated type that lists all the regions allowed to store data. *DataBreachNotification* comes with two attributes, *NotificationEmailAddress* to configure recipient email address, and *TimeToNotification* to configure how long the notification can be delayed after data breach occurs.

Control actions do not necessarily correspond to classical security requirements such as data encryption and decryption, because data assurance defined in GEODAC covers a much broader spectrum than traditional data security. For example, *CertifySafeHarbor* is designed to address privacy requirements at the organizational and business process level, instead of at the data storage level. Different control actions within an assurance capability are not necessarily confined to be at the same level of abstraction, because different control actions from different abstraction levels are assembled to collectively achieve a high-level capability. For example, to achieve data migration requirements, we need *EmailNotification* to notify data owner (process level), and *FingerPrintData* to perform artificial data injection [46] (data processing level).

Note that this vocabulary is extensible to accommodate requirements beyond those in Table 1.

4.2 XML Based Policy Representation

We have extended the WS-Policy framework, based on the GEODAC language constructs introduced in Section 4.1, to form the WS-DataAssurancePolicy language. The mapping for the language constructs identified in Section 4.1 is straightforward. All the non-terminal symbols introduced in Figure 3 have their corresponding XML elements in WS-DataAssurancePolicy. Each control action or assurance capability definition can optionally have an XML attribute called "Namespace" to denote its definition scope. The non-terminal symbols of "attribute definition" and "attribute assignment" in Figure 3 are unified by the XML element called *sda:attribute* (The prefix *sda:* denotes the schema namespace of WS-DataAssurancePolicy). To better partition the XML segments on attribute and capability definitions for each data capability specification associated with a persistent data type, we introduce *sda:IncludedAttributes* and *sda:IncludedCapabilities* to group these attribute definitions and capability definitions.

The XML element *sda:ExactlyOne* represents mutually exclusive relationships among control actions that cannot be chosen by the service customer together. The specification of control action constraints is structured in an XML tree, similar to the one adopted in XACML [27] to specify matching conditions for policy rules. A constraint over the control action's attributes is represented by a *sda:Constraint*, which further contains a predicate represented by *sda:ConstraintPredicate*. The predicate can be hierarchically composed via predicate operators like *and*,

or, *implies*, etc. The leaf predicate is evaluated on selected attributes against predefined attribute values.

Service capability specifications can be packaged in a separate XML document from the customer's data assurance policy specification. In order to establish the necessary document-level linking, in a customer policy specification, we introduce an attribute called *sda:CapabilityPolicyURIs* in the XML element *sda:ServicesSpecifications* to specify the location of the capability specification document as a URI.

The complete WS-DataAssurancePolicy XML schema can be found at [20] as well as the two policy examples introduced in Section 4.1.1 and Section 4.1.2.

4.3 Discussion

Our design choice to restrict one-to-one mapping between persistent data and transient data simplifies the tracking of data ownership (what data belongs to which service customer) in the service environment. It is likely that within a service, there exists data fusion to merge multiple data sources that are tagged with different policies. Our policy framework does not explicitly address such a backend data fusion situation because our focus is on service data processing resulting from direct service customer invocation. A different data assurance process will be required to examine the fused data's policy, and how to merge the involved data policies, e.g., whether to inherit from all of its parents, or be assigned with a different data policy. As an example, the latter could happen when sensitive data has been through a de-identification process and is no longer considered as PII.

The GEODAC specification language does not support formal definition on persistent data types. Instead, we specify an attribute on persistent data to point to the location of formal data definition. A persistent data object can be defined at a different granularity, e.g., a database table, a database row, a file, or an XML document segment. GEODAC is focused on the recording and tracking of the relationship between data objects and data assurance policies, so that the service runtime can respect and enforce the policies accordingly. It is not concerned with data heterogeneousness in the backend persistent store.

5 Data Assurance Console

We provide an interactive *data assurance console* to facilitate service customers to understand the capabilities and specify their assurance requirements. The framework reads the service provider's capability specifications and renders them to a web browser. The console provides two complementary views: *data view* and *policy view*.

The data view lists all published data object definitions for which data assurance capabilities are offered, based on the service capability specification. It allows selection of the data object types that require protection.

The policy view shows the data assurance capabilities for each selected data object type, by retrieving the control action and assurance capability related definitions. This view is used by the security expert to configure the requirements on the data objects of this type. Figure 6

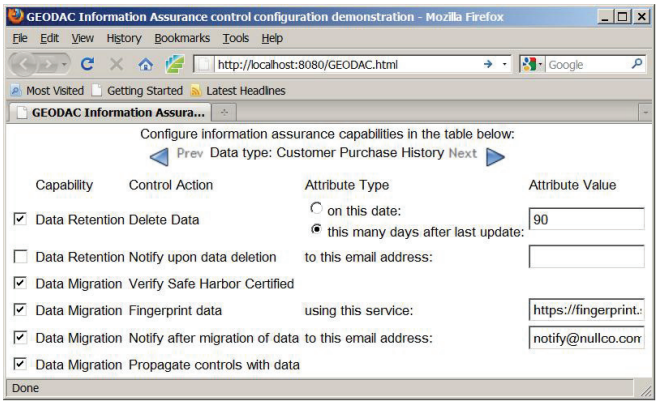

Figure 6. Policy view of data assurance console.

shows a screenshot of the editor showing the policy view for the *CustomePurchaseHistory* data object type. From this view, the security expert may configure the attributes defined in the related control actions. The console generates WS-DataAssurancePolicy for the service customer's policy specification.

6 Enforcing Data Assurance Policies

The customer's data assurance policies are enforceable in the service environment. The enforcement model is encoded in a set of state machines in the service environment that represent the persistent data objects' lifecycle states and the involved state transitions. The control actions for policy enforcement are encoded in the state machine and executed when state transition happens. The policy enforcement approach works as follows:

1) For each persistent data object type and for each data assurance capability, the system designer designs a state machine that represents the lifecycle states and transitions of data objects of the associated type. GEODAC provides a template state machine for each assurance capability. The granularity of a data object can be a file, a database record, database records, etc.
2) The system designer defines a set of primitive actions for each control action and assigns them to state transitions in the state machine.
3) The state machine can be defined hierarchically, i.e., the fulfillment of control actions in a state machine can be further delegated into a child state machine or workflow. Refining an enforcement action into a workflow is useful for scenarios requiring human involvement, e.g., for approval.
4) The data assurance policy (or WS-DataAssurancePolicy) for the persistent data object type contains the configured attributes for each involved control action. These configured attributes are mapped to the corresponding attributes in the state machine. This allows customers to have unique policy configurations applied to the state machine templates.
5) The enforcement component relies on a workflow engine that understands state machine execution and workflow execution. When an event happens, the state machine is activated and the event triggers the state machine to perform corresponding state transition and therefore carry out the policy enforcement actions.

6.1 State Machine as Policy Enforcement Model

Policies are defined at an arbitrarily fine granular level of data. We define the policy enforcement models as state machines that encode the lifecycle stages of data as states 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. One policy enforcement model is defined per data assurance capability. For example, one state-machine model is defined for "data retention" capability. This policy enforcement model describes how to enforce the corresponding set of control actions defined in the data assurance capability, taking into account the configured attributes involved in the control actions. We define a policy enforcement model as follows:

Definition 3 (Policy Enforcement Model). A policy enforcement model *PE* for a data assurance capability is represented as a tuple $PE =< S, s_0, A^E, E, F, T, \mathrm{X} >$ where S is the set of lifecycle states of the data, A^E is the set of actions that collectively enforce the capability, E is the set of events received or generated, $T \subset S \times S \times A^E$ is the set of transitions, s_0 is the initial state, and F is the set of final states. The label of a transition t from state s to s' is an ECA (event-condition-action) rule. Formally, $t =< s, e_i, c, a, e_o, PE_c / W_c, s' >$ is triggered by an event e_i, and if condition c holds, then it may execute action $a \in A^E$, generate event e_o and spawn child enforcement model PE_c or the workflow W_c. X is the set of attributes of the enforcement model, which represents the collection of attributes of actions in A^E.

In general, a control action $a \in A$ of a capability (as defined in section 4.1.1) may refine into primitive actions in A^E. We define the following mapping from the set of control actions into actions in A^E used in the enforcement model.

Definition 4 (Control Action Mapping). A control action $a \in A$ is mapped into a set of actions in A^E from any of A^{PR}, A^{Ev}, A^T, i.e., $\{a' \in A^E \mid a' \in A^{PR} \vee a' \in A^{Ev} \vee a' \in A^T\}$ where A^{PR} is the set of primitive implementation level actions, A^{Ev} is the set of event-related actions, and A^T is the set of time-related actions.

The set of actions in A^E is used in the policy enforcement model. An example of actions in A^{PR} is "Delete Data". Actions in A^{Ev} raise or handle events such as approvals. Finally, actions in A^T are time-related actions such as "Wait" or "Time Delay". The reader may consult Section 6.2.1 for detailed examples of these three action types.

In order to simplify the design of policy enforcement models, we allow hierarchical definition of state machines. This design decision is also consistent with the observation that fulfillment of some control actions may entail complex workflows or action sequences defined on actions in A^E. These may involve long lasting service invocations or human decision such as approval. If the fulfillment of the action in the parent state machine involves performing actions that change the lifecycle stages of data then a child state machine PE_c is introduced. On the other hand, workflow W_c is used when the fulfillment of the action in the parent state machine entails a process involving human interactions.

It should be noted that while taking t, action a, event e_o and PE_c /W_c are optional. A transition from state s to s' may be triggered by the event e_i and action a may be performed as a primitive action without requiring a child state machine or workflow. Indeed, any transition in a state machine, in any level of the hierarchy, can be further refined to a child state machine model. A workflow is considered as a leaf and actions within a workflow are not further refined. We define a workflow as follows.

Definition 5 (Enforcement Workflow). An enforcement workflow is a directed graph (flowchart) with four types of nodes: start, end, action and decision. Formally, we describe a workflow $W =< n_0, n_f, D, N, R >$ where n_0 is the start node, n_f is the end node, D is the set of decision nodes, N is the set of action nodes, and $R \subset A^E \times A^E \cup A^E \times D \cup D \times D$ is the set of transitions. A decision node $d \in D$ has a condition with typically two outgoing transitions, one labeled "*Yes*" (when the condition is true) and another labeled "*No*" (when the condition is false). An action $a \in A^E$ is associated to any action node. An action node can have one or more incoming transitions, but only one outgoing transition.

To summarize, a data assurance policy can be described in a hierarchically composed model, with a state machine at the highest level and the detailed action workflows at the leaves. Note that this model is different than traditional hierarchical state machines [14]. Child workflows are spawned from the parent model asynchronously with their own thread of control to support long delay or human decision. The child workflow notifies its parent model by events. Section 6.2.1 shows an example of parent-child event-based interaction.

Relationships of enforcement model, capabilities and policy specification. For each capability (shown in Table 2), one policy enforcement model is designed which fulfills all the control actions for the capability. A set of policy enforcement templates is associated to each persistent data object type. Each template enables enforcement of a particular capability. The attributes X in the template enforcement model take their values from corresponding attributes in the policy specification *P* for each customer. Indeed, an instance of a policy enforcement model is created from its template by using the attribute values of control actions specified by the customer during the capability configuration phase.

6.2 Enforcement Model Examples

To help understand our policy enforcement approach, 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 set of customers to engage for the campaign. Nullco would like the data retention policy for both data sets 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 2);

(**R2**) After the data is deleted, an email notification is sent to the specified email address in Nullco (*EmailNotification* in Table 2).

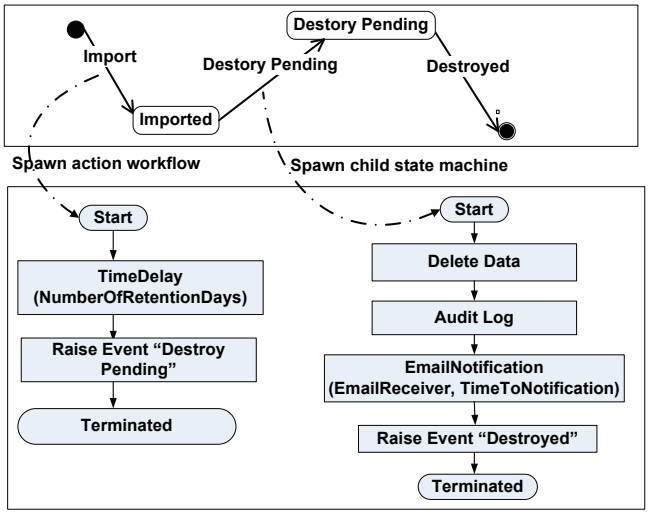

Figure 7. State machine-based data retention policy.

Figure 7 shows the complete policy enforcement model for data retention (note that event names are shown as transition labels in state machines). When the data is *imported*, the event triggers the action of spawning a child workflow (shown by a dashed line), which starts the timer for a delay period that is specified by the attribute *NumberOfRetentionDays* defined in Table 2. 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*. The event also triggers the action in which a second child workflow is launched to perform actual data deletion. An email notification will be sent out by the control action, through *EmailNotification* defined in Table 2, along with its configured attributes, before the child workflow terminates itself. As mentioned in Section 2.3, audit logging is a generic observable control action defined in GEODAC. The step of "Delete Data" is not a control action from Table 2, but is a primitive implementation action used to realize the control action "DeleteBy" (see the action mapping in Section 6.1).

6.2.2 Data Migration

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

(**R1**) The candidate data receiver must be Safe Harbor [38] certified *(CertifySafeHarbor* in Table 2);

(**R2**) A data fingerprinting service [46] must be used to produce a unique copy of the database table for each receiver (*FingerPrintData* in Table 2);

(**R3**) The data owner (Nullco) must be notified upon the completion of migration actions (*EmailNotification* in Table 2);

(**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 2).

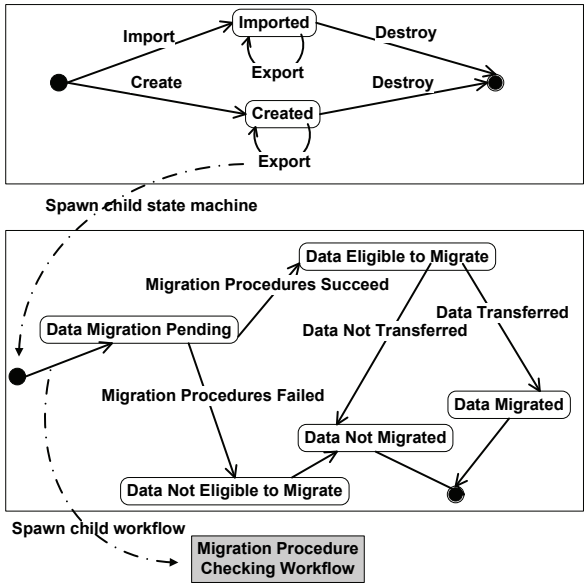

Figure 8. State machine-based policy for data migration.

Figure 8 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 (shown by a dashed line) at the transition to *imported* or *created* due to the event 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 Figure 9) 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 en-

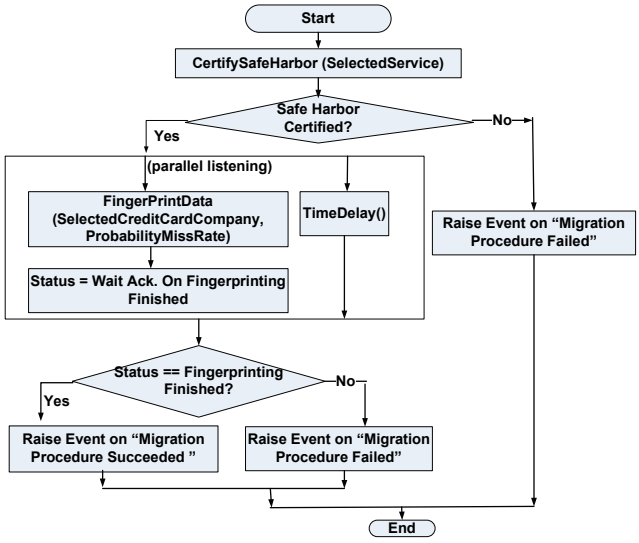

Figure 9. Migration procedure checking workflow.

able notification of the data owner (R3). Note that to realize R4 the policy is sent along with data to the data receiver (a different service provider to which the data is migrated). The policy needs to be enforced by the data receiver in its service environment.

A child workflow spawned due to the transition to the *data-migration-pending* state is shown in Figure 9 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 control actions *CertifySafeHarbor* and *FingerPrintData* defined in Table 2 are involved along with their configured attributes.

6.3 Enforcement Models for Other Capabilities

Section 6.2 shows two examples of data assurance capabilities. In fact, we have expressed all assurance capabilities in Table 2 with state-machine enforcement model except data confidentiality. Data confidentiality could be achieved by mechanisms such as data encryption that do not necessarily require a state machine representation. In this section, we briefly illustrate how to express other data assurance policies identified in Table 2.

Data Appropriateness for Use. To ensure data appropriateness for Nullco's campaign materials we expressed an approval process in a state machine-based enforcement model to require approval of certain actions, such as creation or modification of marketing materials. The approvers can be humans or software services. 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 workflow, which issues requests to solicit approvals. Based on the approval (disapproval) results, the child workflow will raise a *create-approved* (or *create-disapproved*) event. The transition due to the *create-approved* (or *create-disapproved*) event will have the action to accept (or discard) the pending marketing materials respectively.

Privacy Requirements. The control actions *RestrictLocation* and *CertifySafeHarbor* 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* raised from a data store or business process that monitors the associated data. When the *DataBreached* event is triggered, the control actions such as *EmailNotification* will happen.

Data Integrity and Data Availability. In contrast to other data assurance policies such as data retention and data migration, the policy execution for data integrity mainly involves external monitoring from the customer side. More specifically, to verify data integrity, we can design a state machine model to query the outsourced database and perform verification on query results, based on known truth about the outsourced data [48]. Similarly, for data availability, we can follow availability testing strategies such as [18], to periodically test the service and produce an availability report. A trusted third-party could conduct testing on behalf of a service customer.

7 GEODAC Framework: Architecture, Implementation and Evaluation

We have designed and implemented the GEODAC policy management and enforcement framework as described in the following.

7.1 Architecture

The high-level architecture of GEODAC is shown in Figure 10. Customer-specific policy specifications are stored in the *WS-DataAssurancePolicy Repository*. A *data access* event is raised from the application web service, typically from a service interception point defined in a service method. Each service interception point is installed with event adapters to generate the events for the involved data objects. Once a data access event, such as data creation, is sent to the *Policy Management Service* (PMS) from the service interception point, the corresponding policy is triggered by the execution engine, resulting in an activated *policy execution instance*, which realizes the policy enforcement model. Other events that relate to update, fetch, destroy, import, etc. of data are raised and drive state transitions, resulting in execution of different actions defined in the enforcement model.

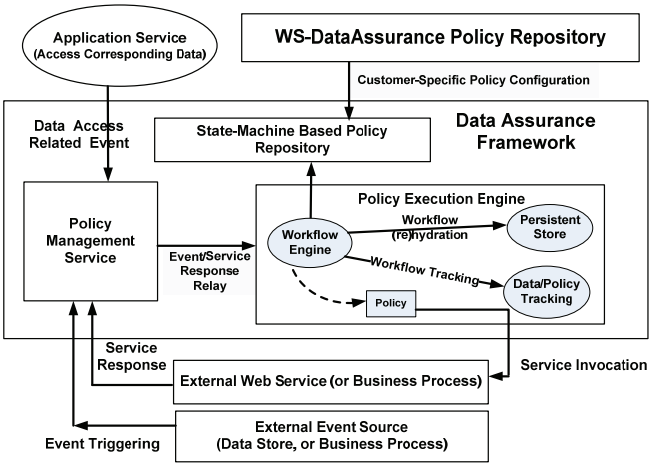

Figure 10. Architecture for policy execution and management.

Event triggering at the service interception point can be customer-specific. If a customer opts out of the involved data object's assurance policy, the event will not be triggered. The policy execution instance is also customer-specific. The activated policy is based on the state machine template stored in the *State Machine-Based Policy Repository*, and configured by the attributes defined in WS-DataAssurancePolicy.

External web services or business processes can trigger events as well. The triggered event is delivered to the PMS, from which the event is routed to the target policy execution instance.

The policy execution instance can invoke external web services, e.g., to verify the Safe Harbor certification. Synchronous responses are returned to the policy execution instance whereas asynchronous responses are returned indirectly through the PMS.

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 [39] such that the activated policy execution instance only resides in memory when an event-triggered transition occurs. As mentioned earlier, events may be raised from child enforcement models, or from an event source such as a data store or a business process.

The PMS is the façade for policy execution engines. To achieve scalability in a machine cluster, each policy engine runs on a particular machine and registers itself to the PMS. The PMS receives messages from either service interception points or external web services/business processes, and deposits the messages to the message queue that belongs to an active execution engine. Correspondingly, the chosen execution engine pulls the message from its message queue, and its message dispatcher translates the message into a local event. The event is then raised to the targeted policy execution instance. Thus any cluster machine can handle any events or messages. However, to guarantee total ordering of messages dispatched to the same target policy execution instance, the PMS's routing protocol ensures that:
(i) All the messages that belong to the same policy execution instance are routed to the same policy execution engine; and
(ii) If child enforcement models of a policy execution instance are involved, the child model related messages are routed to the same policy execution engine that handles the policy execution instance.

7.2 Implementation

We have prototyped the WS-DataAssurancePolicy enforcement framework described in Section 7.1 in Microsoft .NET environment. The service modules, i.e., WS-DataAssurancePolicy Repository, Policy Management Service, and Data/Policy Tracking Service are all implemented as web services and hosted in different machines in the cluster. The message queue is implemented as a .NET remoting object server on each machine from which the local policy engine can pull the messages. Correspondingly, the Policy Management Service can push the messages to the selected queues in the cluster.

A state machine policy is encoded in Microsoft Windows Workflow Foundation (WWF) [39]. A top-level state machine is described as a WWF *state machine workflow*. If a child enforcement model is a workflow that represents an action sequence, it is described as a WWF *sequential workflow*. If the child model is a state machine, it is described as a WWF *state machine workflow*.

In a policy execution instance, the parent workflow invokes a child workflow through the WFF workflow activity pattern called *InvokeWorkflow*. A communication mechanism in WWF called *Local Communication Services* (LCS) is employed 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.

All the policy configuration related attributes defined in WS-DataAssurancePolicy have their counterparts defined as *Dependency Properties* in the topmost state-machine. The policy configuration attributes and the Dependency Properties need to be one to one mapped. The mapping happens when the policy execution engine creates the topmost state machine instance. Such a data binding process allows policy configuration attributes to be passed down to the state machine hierarchy.

Finally, policy execution requires tracking of the parent/child relationship between child enforcement models and their parents. In addition, our framework also supports correlation identifiers [26, 39] to track application-level relationships. The Data/Policy Tracking Service facilitates both functionalities.

7.3 Experiments with Sample Policies in GEODAC

We have realized data retention and data migration, the two policies examined in Section 6.2, in WWF. In our experimental setup, a machine in the cluster is devoted to host all the web services shown in Figure 10. 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 enforcement models are distributed to each engine. The prototype demonstrates the feasibility of enforcing customer-specific data assurance policies using the GEODAC framework.

8 Related Work

There is much work that proposes languages and support to express and enforce message-level security policies in Web services (see [5][9][12][32][41] for surveys). We discuss related work in data assurance policy specifications and data assurance policy enforcement.

Data assurance policy specifications. Existing work proposes Web services security policies [1][5][9], with focuses on message level security requirements such as confidentiality, integrity, authentication and trust [41], or resource protection through access control [7][8]. Standardized policy languages such as WS-SecurityPolicy [28] and XACML [27] fall into the same category.

P3P focuses on privacy policy for Web sites [47]. Privacy concerns in our work are focused on enterprise needs and in the context of outsourcing. The work of [13] presents abstractions for expressing privacy concerns at the business protocol level. Our framework is not limited to expressing only privacy concerns. In addition, we provide policy enforcement mechanisms.

Some recent work presents high level frameworks for security requirements (e.g., [32]) and at the business proc-

ess level (e.g., [17][23]). However, these works focus on message-level security polices and do not enforce policies for persistent data.

Data assurance policy enforcement. In the service oriented environment, there exist two policy enforcement points: service messaging and backend processing. The message-level enforcement has been well investigated [28][41]. Our data assurance policy and its enforcement are focused on persistent data objects that are stored, processed and migrated in outsourced service environments. Access control provides enforcement at backend processing. There are works (e.g., [8]) that focus on access control mechanisms in outsourced services environments which are complementary to our work.

Another complementary theme of work is distributed usage control [33], which discusses handling of sensitive data passed from the data provider to its consumer through obligations. In our data assurance framework, detailed obligations are expressed as the child enforcement model attached to specific transitions. Our solution provides a practical approach to enforce and propagate obligation policies in outsourced service environments. From a broader view, usage control is a generalization of access control to cover authorizations, obligations, conditions, continuity and mutability [30].

It is possible to adopt usage control models for some of our data assurance requirements such as data appropriateness for use, data retention, and a subset of privacy requirements. However, notification in our privacy requirements and data retention does not deal with data access directly. Other data assurance requirements such as data confidentiality cannot be easily mapped to data usage control. Furthermore, our customer-specific policy enforcement model and mechanisms support enforcement actions that are event-driven, or involve long lasting service invocation or human decision processes. To the best of our knowledge, the modeling and enforcement of these aspects are not addressed by usage control models.

State machines have long been used to describe and support system behaviors. Trust-Serv uses state machines to express trust negotiation policy models [42]. However, it focuses on external interactions to establish trust. PRIME [34] supports privacy obligation management, with the focus on expressing and enforcing data retention policies. A policy engine is presented in [6] to handle obligation-related policies through dynamic policy construction from policy templates. In contrast to the above works, our data assurance policy framework offers broader data assurance aspects, more flexible policy customization, and a customer-specific policy enforcement approach in outsourcing environments.

Our work provides application-level data assurance capabilities. With respect to cloud computing, hardware-platform-as-a-service, e.g., Amazon.com, focuses on hardware virtualization and is not aware of business data requirements. Data encryption is the only recommended mechanism to protect business data in persistent stores such Amazon.com's Simple Storage Service [1], which as we have shown is not sufficient for broader data assurance categories identified in this paper. Salesforce.com, a platform-as-a-service provider, currently provides only role-based access control for data assurance [37].

Compared to other works that are focused on security policy and enforcement in distributed workflow management (e.g., [2, 24]), our policy framework defines enforcement actions, which are incorporated into workflows that are applied on the lifecycle of data while in service environments. We use workflow as a tool for expressing and enforcing data assurance policies. However, in the works on enforcing security policies in distributed workflow environments [2, 24], the focus is on securing workflow execution. Indeed, the involved security mechanisms are concerned with securing the information flow of a workflow within and across organizations. Nevertheless, such workflow management environments do share similar security concerns with GEODAC, e.g., traceability and integrity [24] and access control [2].

9 Conclusion and Future Work

In this paper, we have presented the GEODAC framework including a methodology, language, and mechanisms for identifying, capturing and enforcing data assurance policies. To the best of our knowledge, this is the first work which characterizes data assurance requirements in outsourced services environments, and proposes a mechanism for enforcing these policies to protect persistent data stored in service provider environments.

We have proposed the GEODAC data assurance language, its grammar and its XML-based representation called WS-DataAssurancePolicy, to allow service providers to specify the data assurance capabilities they provide on persistent data, and service customers to configure these capabilities. The language allows binding the policies to data items at various levels of granularity. We prototyped the enforcement mechanism for data assurance policies based on state machines and workflows. In cases where service customers cannot directly specify enforcement actions, the framework allows defining observable actions to provide confidence of policy compliance.

The GEODAC framework opens new avenues for future work. The framework is designed to enable the propagation of data assurance policies with data when data migrates to sub-contractors, in which subsequent policy enforcement occurs. We are planning to extend the enforcement framework for distributed enforcement of data assurance policies (e.g., in sub-contractor environments). Another next step is to evaluate framework scalability in a larger context (100 machines and beyond).

References

[1] Amazon.com, *Amazon Web Services: Overview of Security Processes*, http://s3.amazonaws.com/aws_blog/AWS_Security_Whitepaper_2008_09.pdf.

[2] S. Ayed and et. al., "Deploying Security Policy in Intra and Inter Workflow Management Systems, "ARES 2009, pp. 58-65.

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

[4] A. Baldwin and S. Shiu. "Enabling shared audit data". Int. J. Intf. Secur., 4(4): 263-276, 2005.

[5] 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.

[6] M. Casassa Mont, F. Beato, "On Parametric Obligation Policies: Enabling Privacy-aware Information Lifecycle Management in Enterprises," IEEE POLICY 2007, pp.51 – 55.

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

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

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

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

[11] M. Fratto, "Internet Evolution – Cloud Control," Information Week, Jan. 2009.

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

[13] R. Hamadi et. al., Conceptual Modeling of Privacy-Aware Web Service Protocols, CAiSE 2007.

[14] D. Harel, Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8(3):231–274, June 1987.

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

[16] HIPAA, http://www.hhs.gov/ocr/privacy/index.html

[17] M. Jensen, S. Feja, "A Security Modeling Approach for Web-Service-Based Business Processes," IEEE ECBS 2009, pp.340 – 347.

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

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

[20] J. Li et. al., "A Data Assurance Policy Specification and Enforcement Framework for Outsourced Services," HPL Tech. Report 2009-357.

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

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

[23] M. Menzel et. al., Security Requirements Specification in Service-oriented Business Process Management, ARES 2009.

[24] F. Montagut, R. Molva, "Traceability and Integrity of Execution in Distributed Workflow Management Systems," ESORICS 2007.

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

[26] OASIS, WS-BPEL 2.0, 2007.

[27] OASIS, XACML, Version 2.0, Feb. 2005.

[28] OASIS, WS-SecurityPolicy 1.2, July 2007.

[29] Object Constraint Language (OCL), OMG , Version 2.0, May 2006.

[30] J. Park and R. Sandhu, "The UCON ABC usage control model," ACM Trans. on Information and Systems Security 7 (2004), 128-174.

[31] PCI DSS (1.2), PCI Security Standards Council, Oct. 2008.

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

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

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

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

[36] P. Resnick et. al., "Reputation systems: facilitating trust in Internet interactions," Comm. ACM, 43(12):45-48, 2000.

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

[38] Safe Harbor, http://www.export.gov/safeharbor/index.asp.

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

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

[41] 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.

[42] H. Skogsrud et. al., "Trust-Serv: Model-Driven Lifecycle Management of Trust Negotiation Policies for Web Services," WWW 2004.

[43] Statement on Auditing Standards No. 70 (SAS 70), AICPA.

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

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

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

[47] W3C-Platform for Privacy Preferences, http://www.w3.org/P3P/.

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