Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2013, Article ID 298037, 13 pages
http://dx.doi.org/10.1155/2013/298037

Research Article

Accountability in Enterprise Mashup Services

Joe Zou[1] and Chris Pavlovski[2]

[1] *Centrin Data Systems, 1 Boxing 8th Road, Beijing 100176, China*
[2] *IBM Global Business Services, 348 Edward Street, Brisbane, QLD 4000, Australia*

Correspondence should be addressed to Chris Pavlovski; chris_pav@au1.ibm.com

Received 16 September 2012; Accepted 17 December 2012

Academic Editor: Xiaoying Bai

As a result of the proliferation of Web 2.0 style web sites, the practice of mashup services has become increasingly popular in the web development community. While mashup services bring flexibility and speed in delivering new valuable services to consumers, the issue of accountability associated with the mashup practice remains largely ignored by the industry. Furthermore, realizing the great benefits of mashup services, industry leaders are eagerly pushing these solutions into the enterprise arena. Although enterprise mashup services hold great promise in delivering a flexible SOA solution in a business context, the lack of accountability in current mashup solutions may render this ineffective in the enterprise environment. This paper defines accountability for mashup services, analyses the underlying issues in practice, and finally proposes a framework and ontology to model accountability. This model may then be used to develop effective accountability solutions for mashup environments. Compared to the traditional method of using QoS or SLA monitoring to address accountability requirements, our approach addresses more fundamental aspects of accountability specification to facilitate machine interpretability and therefore enabling automation in monitoring.

1. Introduction

The recent and rapid expansion of Web 2.0 has considerably placed pressure upon industry to institutionalize new technologies and conform to emerging standards. While agreement on the scope of the term Web 2.0 does vary, O'Reilly provides a commonly accepted definition, noting this to include a range of enhanced services including web services, wikis, blogging, BitTorrents, and syndication [1].

The rapid growth of Web 2.0 has also introduced a number of new design patterns and architectural styles in web development. One of the notable techniques involves mashing up information from existing services to deliver new value-added services. This process effectively involves the drawing of content from several sources to create a new content or service. The resulting web page is then referred to as a mashup of the existing content.

While mashup services bring flexibility and speed in delivering new valuable services to consumers, the legal implications of using this technology are significant. Researchers in law conclude that the development of mashup services is fraught with potential legal liabilities that require careful consideration [2].

The issue of accountability associated with the mashup practice remains largely ignored by the industry. Current formal practices suggest that the mashup developer and original content source owner disclaim any warranties [2]. This appears to be temporarily acceptable since most services from Web 2.0 sites are free to internet users. This means that as long as consumers accept the terms and conditions, the issue of accountability is largely avoided. Notwithstanding, as these services mature to involve some payment, such an approach may no longer be tenable to all parties.

Traditionally, accountability implies that an entity has an obligation for the execution of authority and/or the fulfilment of responsibility [3]. Nonrepudiation of transaction is also a major requirement for a service requester and a service provider. However, in a mashup service scenario, the issue of accountability is more complicated. Firstly, there may be several implicit service providers involved due to the fact that the service is mashed up from a number of sources. Secondly, the content presented may not be delivered by the content

originator. Furthermore, the sourced content may be altered or extended during the mashup process. Considering this problem further, does the body who modifies or augments the content assume entire liability for all the repurposed content, including all accuracies and inaccuracies?

This paper is an extended version of earlier work that appears in [4], in particular, providing an implementation scenario. In this paper we consider that the accountability issue in mashup services is a broader and more complex theme when compared to nonrepudiation in an eCommerce transaction. We propose a framework that includes the service or content creator as well as the new owner of the resulting mashed-up service. While accountability issues may not be fully addressable with the current technology, we believe that the first step towards enabling accountability in mashup services is to add more disclosure, trust, and undeniability. This includes identities of all the parties involved and traceability in service composition. We also suggest that the concepts of involved parties and roles are essential in the service ontology model, such as in the Ontology Web Language for Web Services (OWL-S) model [5].

Given that accountability in mashup has not been treated rigorously before, we view the main contributions of this paper as follows.

(1) The underlying mashup accountability issues in practice are analysed with a formal definition for accountability in mashup solutions provided.

(2) A framework and ontology are proposed to model accountability in mashup environments.

(3) An implementation scenario is outlined that applies these methods in practice.

Using these methods, it is hoped that more effective accountability solutions can be prepared on the basis of our framework and model. The remainder of this paper is structured as follows. Section 2 provides background information on the mashup paradigm and related work in the current literature in accountability. In Section 3 we suggest a definition for accountability to address the additional requirements of mashup services. This is followed by a framework and ontology that may be used to model accountability. In Section 5, we illustrate how the framework and ontology can be applied in an implementation scenario. Finally, we summarize our results and observations, discussing areas of further work.

2. Related Work

2.1. Overview of Mashup Services. The term mashup originates from the practice of mixing song samples from two or more sources to produce a new sound track. In the context of the Internet, mashups are websites or applications that combine content from more than one source into an integrated application. This is generally achieved by using third party content provider application programming interfaces (APIs) or open technologies, for instance, Ajax and PHP, and syndicated feeds such as RSS or ATOM. In addition, since the content may be obtained from several sources, intermediate businesses have emerged that act as content brokers. These intermediaries provide access to several content sources from 3rd parties and also supply functions to support the mashup process.

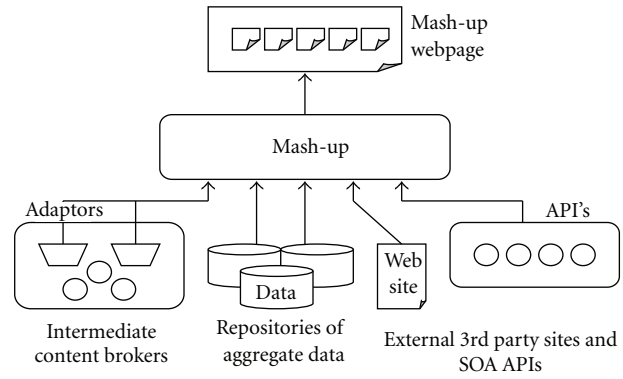

Figure 1: Mashup services.

Based on the concept of service composition in Service-Oriented Architecture (SOA), mashup provides flexible and dynamic services with rich experience. This technology also enables a dynamic form of service reusability in contrast to the traditional method of static "cut and paste" reusability. However, since mashup involves the aggregation of another party's content into some new service or application, a number of legal issues are introduced [2]. When legal issues arise, accountability will become the critical concern for the parties involved.

Figure 1 illustrates the fundamental concepts in Web 2.0 mashups, where data and content are drawn from a range of sources to produce a new aggregated content or service. For example, the content may be drawn from local data repositories, from existing local and external web pages, accessed via SOA-based APIs, and from intermediate content brokers.

While mashup applications and services are growing at a rapid rate, currently they appear to be applied in nonmission critical services and are offered to the internet consumer largely as free services. In practice, legal responsibilities are generally avoided by the content provider disclaiming all warranties and liabilities [2].

More recently, industry leaders are accepting mashup as an enterprise tool to enable the creation of so-called situational applications. These types of applications solve business problems such as inventory management, sales, and marketing information management [6]. This emerging approach has been termed "enterprise mashup" and several enterprise tools have been released [7]. Enterprise mashup may be viewed as a Web 2.0 technology that builds upon the flexibility offered by SOA, and having a requirement for increased security.

Although enterprise mashup services hold great promise in delivering a flexible SOA solution in a business context, the lack of accountability in current mashup solutions may render this ineffective in the business environment. As such, as more enterprises embrace this technology in building IT

solutions, the issue of accountability will manifest as a key concern for the service stakeholders.

2.2. Related Work on Accountability. The meaning of the term accountability appears to vary considerably and is dependant upon the context. Traditionally the topic of accountability has attracted much interest with focus on the eCommerce transaction. According to Kailar, accountability is "the property whereby the association of a unique originator with an object or action can be proved to a third party" [8]. The definition implies nonrepudiation in an eCommerce transaction. Kailar also proposes a framework for the analysis of communication protocols that require accountability [8, 9].

Bhattacharya and Paul assert that while a digital signature can provide help in enabling accountability in two direct communication nodes, it cannot fully address the accountability issues in multihop message due to the sender's ambiguity problem [10].

In [11], the scope of accountability is broadened to represent the ownership of the responsibility to meet requirements in an end-to-end business process. The authors propose "Accountability Centered Approach" (ACA) for business process engineering. The ACA approach suggests iterative decomposition of accountability to appropriate levels and mapping of subaccountabilities into activities.

A 3D approach in accountability modelling (detect, diagnose, and defuse) is proposed in [12] to discover and eliminate the root cause of problems when violations of service level agreement occur in business processes. The approach adopts Bayesian network reasoning for root cause analysis and service reputation model to address problematic web services.

While existing research on accountability helps traditional eCommerce application and SOA business applications, the issue of accountability in service mashup has not been treated in the literature. In addition, Gerber reviews the implications of using mashups and points out a number of legal issues [2]. This includes copyright misuse, trademark violations, false advertising, contract law issues, patent infringement, warranty, and the rights and privacy of individuals. The author also observes that these legal issues require consideration prior to design or implementation of mashup applications. These issues further motivate the need to address accountability for enterprise mashup services.

Eriksén comprehensively explores the notion of accountability for information and communication technologies [13]. The author cites a general definition of the term accountability as follows: "responsible for giving an account (as of one's acts), answerable," or "capable of being accounted for" [13, 14]. From the analysis of three key articles, the author observes that a common theme prevails in software engineering, which is the characteristic of "making visible and accountable," and also poses the question of "accountability for whom?" These observations further support the notion that accountability for mashup service has a requirement to disclose roles, responsibilities, and current transaction state. Furthermore, Johnson and Mulvey analyze relationships and responsibilities outside the developed IT systems [15]. They focus upon the accountability of system designers with respect to clients, users, and those affected by decision systems, prompting the question "are system designers responsible for the outcomes that result from use of their systems?" Our work addresses visibility through disclosure and identifies for whom the accountability is intended for, with defined roles. In addition, the responsibility aspect is also treated as a key element that requires consideration in accountability.

3. Definition of Accountability

In [3], it is suggested that the term "accountability" is an often used word with no common definition that can be found. The special interest group authors [3] also conduct extensive research of the literature and have provided a definition of accountability in the context of service performance, see Box 1.

We observe that this definition requires strengthening in the multiparty scenario such as mashup service environments. In addition to nonrepudiation, managing trust is also important for entities to collaborate [16]. Moreover, we wish to strengthen this definition with nonrepudiation and trust with one or multiple entities.

In moving towards a definition, we first propose that the essence of accountability involves four elements from an IT perspective, refer to Figure 2. In the original definition of Box 1 a person, group, or organisation can be translated to the concept of identity, whereas execution of authority implies the concept of role. According to Certo, responsibility is an obligation that someone "accepts" and is not allowed to delegate or pass on to someone else [17]. Accepting implies that there is some form of agreement in place. "Answering" and "reporting" relate to disclosure. Assuming liability for results requires a way to clearly demonstrate who has done what. The term assuming liability may be viewed as ambiguous, and considering that trust may vary considerably in a multiparty environment, this needs to be strengthened in order to remove plausible deniability, (i.e., introducing nonrepudiation).

The elements of Figure 2 involve the identity of the involved party, the role the party plays, and the agreed responsibilities in the form of contract, agreement, or signed-off requirements. The last element is the performance outcome, the evidence of who has done what.

We now use a mashup example to demonstrate these accountability elements, see Table 1. In the scenario, Entity B offers a security trading platform to allow their customers to trade various securities globally. It has contracts with different real-time financial data providers to provide price data, which is fed into a charting application provided by a service provider to produce price charts. For a particular trading transaction, customer Alice initiates the trade request with Entity B. This is based on the pricing chart provided by Entity C's charting service, with real-time price input from Entity D.

Using this IT services example, properties of accountability also imply the following:

Accountability refers to the obligation a person, group, or organization assumes for the execution of authority and/or the fulfillment of responsibility. This obligation includes the following:
(i) answering—providing an explanation or justification—for the execution of that authority and/or fulfillment of that responsibility,
(ii) reporting on the results of that execution and/or fulfillment, and
(iii) assuming liability for those results.

Box 1: Accountability for performance.

Table 1: Accountability roles in practice.

Identity	Role	Responsibility	Outcome
Alice	Trade requestor	Enter code and bid price. Provide funds for purchase.	The request accepted by Entity B.
Entity B	Trade provider	Display result page with data from Entity C and D. Execute trade requested. Pay Entity C and D fees due.	The trade is executed. Fund transferred from Alice's account.
Entity C	Charting service provider	Provide correct charted pricing indicators.	Chart is displayed and the fee is received from Entity B.
Entity D	Real-time price provider	Provide real-time pricing with integrity.	Data feed is provided and receive fee from B.

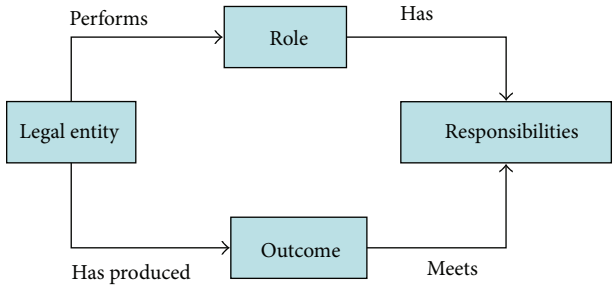

Figure 2: Accountability Elements.

(i) clear disclosure of the roles, responsibilities, and transaction status by all parties;

(ii) each party dutifully carry out their obligations;

(iii) there exists readily available evidence of the services rendered;

(iv) the involved parties cannot repudiate services rendered.

These properties reenforce the need for trust and nonrepudiation. However, in a legal sense, an automated IT system is not a legal entity that is accountable. Rather, it is the person, group of people, or company which is legally accountable. In the context of multiple entities involved in a mashup service, we now provide a more formal definition for accountability by extending the definition in [3].

In [3], responsibility is the obligation to perform, while accountability is the liability that one assumes for ensuring that an obligation to perform is fulfilled [3, 18]. In addition, the term authority is the right to act without prior approval from higher management and without challenge from managing peers [3, 18]. The authors point out that authority is assigned, while responsibility is delegated. This implies a top-down decomposition of authority. Given the bottom-up method of building mashup services, this definition may not strictly apply. Rather, responsibility and authority must be sought and agreed upon between all peer content or service providers, rather than delegated. As pointed out in [17], responsibility is an obligation that is accepted; hence we observe that agreement should be sought. Finally, trust may be established among peers through evidence based on historical behavior and past interactions [16]. Considering these points, we outline the extended definition, by strengthening the definition with multiparty trust and nonrepudiation, making this binding to several parties, see Box 2.

This definition is applicable to both the multiparty service environment (such as mashup) as well as the single party service provider. We also note that the last point of this definition uses the term trusted which also implies that all entities are authenticated. Hence, the accountable service provider would naturally maintain some form of a binding registrar that identifies the subordinate accountabilities present. In order to satisfy this, the approach in [11] would seem to naturally satisfy this condition.

In light of the example and the objective to strengthen the term accountability for the broader context of multiple parties, we observe these additional properties.

(i) Trust: authentication of identities and agreement of accountability between all entities with evidence of behavior.

(ii) Nonrepudiation: undeniable liability with full disclosure (evidence).

4. Service Accountability Framework

Building upon the definitions in this previous section, we now propose a framework, as a metamodel and ontology, for modeling solutions in accountability for the mashup domain.

Accountability in services refers to the obligation that several persons, groups, or organizations assume for the execution and fulfillment of a service. This obligation includes:
(i) Answering, providing an explanation or justification, for the execution of that authority and/or fulfilment of that responsibility;
(ii) Full disclosure on the results of that execution and/or fulfillment;
(iii) Undeniable liability for those result (non-repudiation); and
(iv) Obtaining trusted agreement of accountability from all entities involved in the service, who in turn are bound to the obligations set out above.

Box 2: Accountability for multiple parties.

The metamodel focuses upon the roles and responsibilities from an information systems perspective and is intended for IT developers. The ontology focuses upon the liabilities and agreements aspect of the definition which is useful to establish the contractual terms and definition between the respective parties.

The current literature in IT has placed much focus on the identity and performance outcome elements, which are the most difficult issues to address as that involves trust and nonrepudiation. Security frameworks such as PKI alone cannot address the issues of trust in this computing environment, rather a robust security process framework and security protocol are necessary [19, 20]. As pointed out in [10], digital signatures by themself do not solve the nonrepudiation issue in a multiple party environment due to the sender ambiguity problem (i.e., a party can deny receiving a message by accusing the nonperformance of the intermediary node).

While the identity and performance outcome are essential elements in the accountability framework, the role and responsibility elements are equally important. In fact, we argue that disclosure on the role and responsibility elements is the first step towards an accountability solution. This is because without a clear understanding of the roles and responsibilities by each involved party, the outcome and entity accountable can be disputed.

In a mashup service scenario, the service requester may send a request to a mashup service provider, who in turn forwards the request to the source service provider(s), before aggregating this into a new form for presentation. The issue to observe is that the original service requester may not know the identity of the original service providers. On the other hand, the original service provider is also not aware how their content may be used by the mashup service provider. This motivates the need to find an approach to enable disclosure of roles and responsibilities in mashup services, especially for the enterprise mashup services environment, that are mutually acceptable to all parties involved, whether directly or indirectly. For instance, source content providers may have restrictions on how their content may or may not be used.

We propose a framework for modeling the behavior of mashup services based on SOA and hence briefly visit the fundamentals of this archetype. It is commonly agreed that SOA is an architectural style that involves a triangular relationship amongst three entities: service requester, service provider, and the service registry [21, 22], see Figure 3.

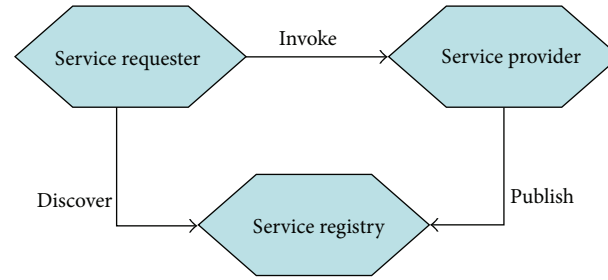

Figure 3: SOA Architectural Style and Actors.

While the model captures the essence of the service-oriented architectural model, it may fall short on enabling accountability in service-oriented architectures in the mashup service environment. For instance, this does not address the roles of multiple parties and the associated responsibility of disclosure and nonrepudiation.

In a mashup context, it is important to note that there are multiple service providers involved. There is also the introduction of service source as a separate entity to the provider, although, in some cases the service provider is the same entity as the service source. In practice, the service provider may engage several external content source parties to participate in constructing the service. In this situation, the service provider relies upon the source for accuracies of supplied content. As suggested in [2], there are a number of legal issues that need to be considered prior to developing mashup applications. As such, both the service provider and source are required to assume responsibility to ensure that the mashup service complies with the intended application (and defined terms and conditions).

Disclosure of roles and responsibility, to a large extent, can be enabled by rich service metadata and facilitated by functions provided by the service broker; rich service metadata means adding semantics to allow machine interpretation and reasoning. Currently, the registry (UDDI) provides service metadata in terms of business entities, taxonomy, and reference to service information. The registry is a dynamic name binding service that is syntax based [23]. However, in mashup several sources require identification and these may need to be trusted sources in an accountability sense. We wish to enable semantic meaning, as in OWL-S [23], and suggest a more sophisticated role to facilitate trust by enabling richer metadata to capture aspects such as

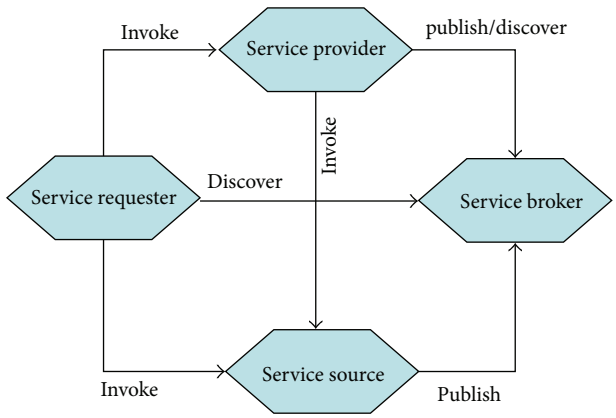

Figure 4: Mashup role responsibility.

traceability of service composition and responsibilities for several parties. Using this extended metadata, the service request may be appropriately associated with source content that addresses both the requirement for content and the need for accountability. In some cases, untrusted content will suffice; in other situations such as enterprise mashup full accountability will be necessary.

4.1. Roles and Responsibilities Metamodel. Based upon the previous discussions, we argue that the two identified roles, service requester and service provider, do not adequately represent all the roles involved in mashup service interactions. We now propose a model to depict these relationships. The revised model is shown in Figure 4. This is composed of the additional roles: service source and service broker. The service registry is still implied in this model, residing with the service broker.

Note that the service provider is a special type of role in the mashup environment which plays as both the requester and provider at the same time. The service provider will draw upon several internal and external sources and provide a resultant mashup page to the service requester. When sourcing content from a broker or service source, the provider acts as the requester. The service source publishes a single or discrete set of (common) content sources that may be accessed directly by the service requester or can be built upon and merged with other content source by a mashup service provider.

As pointed out in [2], new intermediary businesses have emerged that aggregate and broker content from several sources, in essence becoming a one stop shop of various content sources for mashup service providers. The broker supplies content to a service provider who is in turn able to mashup and repurpose the content for a service requester. This means that the service requester may discover services from the broker and invoke this from a service provider. Both the service source and broker publish their available services.

The service broker provides several additional benefits: as a trusted brokering agent (notary for unknown sources), monitoring (audit trail and evidence) to address the disclosure and nonrepudiation requirements, rating functions, and managing a combined registry and repository for multiple sources. Hence, the service broker role can be further refined into detailed roles based on these intermediary functions performed, see Figure 5. The service requester in SOA does not necessarily imply that the entity is a user. This actually refers to the client of the service, which may be another application service or software agent. In an enterprise environment, the participant role in SOA normally represents an organization or party.

The enhanced role interaction model caters for both mashup and traditional service oriented architectures. This helps to understand and define the roles and responsibilities in service metadata. Thus the involved parties and their roles and responsibilities can be discovered and interpreted at runtime and therefore achieve the purpose of disclosure. This model is useful to information systems developers, helping them to identify roles (entities) and responsibilities in an accountable mashup services solution.

4.2. Liabilities and Obligations Ontology. The previous section focused on the roles and responsibilities in a mashup environment, outlining a model from an information systems perspective. This section models the liabilities and obligations from a legal and contractual perspective. This will assist in preparing the engagement basis and contract documents, by identifying the legal entities and artifacts that require consideration when preparing agreements to ensure accountability.

The proposed high-level accountability ontology is illustrated in Figure 6. In this ontology framework, a person or organization is a legal entity that has an identity. A legal entity enters into agreement with other legal entities. The agreement embodies rights and obligations. Rights entail considerations and also imply entitlement for damage if considerations are not met. The obligation sets out the requirements that need to be delivered and penalties if the requirements are not met.

Assigned with the required authority, the legal entity (Figure 6) takes some role, which executes tasks to deliver the requirements. In the context of accountability, the task class has two subclasses, one is service task which provides the intended service; the other is the accountability task which includes disclosure of authority, outcome, and evidence.

OWL-S is the commonly accepted web services ontology language that provides a core set of markup language constructs to describe web service in an unambiguous, machine interpretable form [24]. Thus it will be a natural approach to use those constructs to define the accountability elements in the service metadata. Using the general accountability model in Figure 6, we combine the high level service property constructs from OWL-S [4, 23] (service class and then its property classes: service profile, service model, and service grounding) to address the mashup environment. The extended accountability ontology framework is thus illustrated in Figure 7.

In the context of enterprise mashup environment, a legal entity enters into an agreement with other legal entities in order to participate in a service arrangement through web service interactions. The agreement enables the legal entity to assume a specific web service role to fulfill the obligations

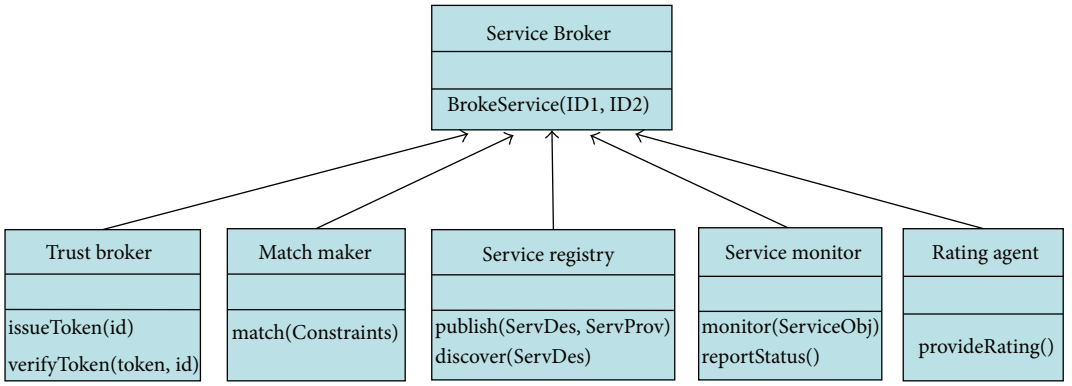

Figure 5: Expanded roles and responsibilities.

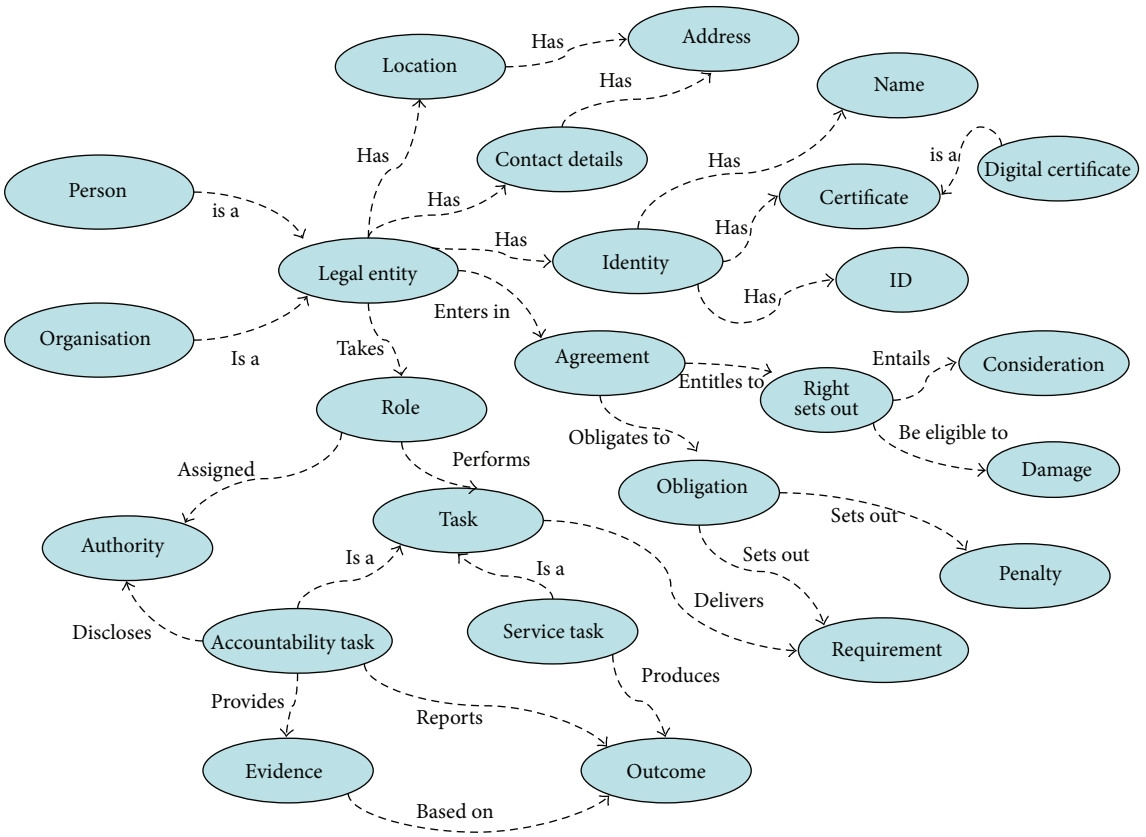

Figure 6: General accountability ontology.

while receiving the considerations. As illustrated in Figure 4, the specific web service role can be service requester, service source, service provider, and service broker. This role will carry out tasks to deliver the requirements setout within the obligations. The role performs a task which has two aspects: one is the normal web service and the other is the accountability task. The accountability task includes disclosure and reporting. Disclosure in this context means disclosure of service metadata, providing evidence of the service outcome. Service metadata may include identities of the involved legal entities, roles that they play, reference to the service agreement, and reference to the original content in the case of mashup service. Service agreement includes terms and conditions of the service.

5. Implementation Scenario

In this section, we demonstrate how to apply the accountability framework and ontology during the development lifecycle of a mashup solution. The example scenario involves a travel agent intending to develop a website that provides travel booking services to online customers. The site will mash up mapping data from a 3rd party together with travel location information in order to present tour routes and destinations

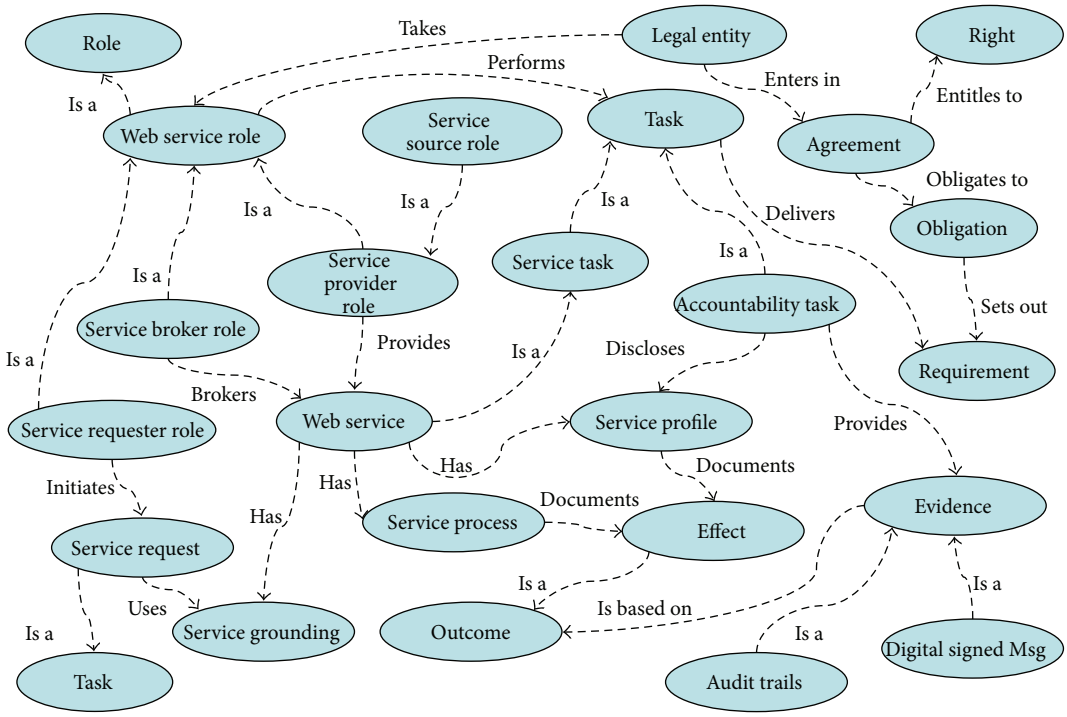

Figure 7: Mashup accountability ontology.

on a consolidated map. The metamodel will assist during the software development lifecycle in identifying the roles and responsibilities. The ontology will assist in the preparation of contracts and agreements between the various entities.

5.1. Roles and Responsibilities in Solution Development. A customer accessing the website is able to select a desired travel plan and confirm a booking. The site will automatically book the air tickets and hotels through the mashed-up APIs provided by independent external airline and hotel businesses. Finally, the solution will also include an electronic commerce transaction to accept payments. Figure 8 illustrates the source and method by which these individual services are pulled together to produce an enterprise mashup service.

The first step in the analysis is to identify the roles and responsibilities associated with each participating entity in the mashup service. During architecture and design, these roles and responsibilities are expanded and mapped to design components, which are subsequently implemented in software. These lifecycle phases are now described.

5.1.1. Requirement Gathering: Allocating Roles and Responsibilities. Using the roles and responsibilities metamodels from Figures 4 and 5, the mashup service environment is systematically analyzed to identify the specific roles, responsibilities, and expected outcomes for each entity participating in the solution. We conduct an analysis on the service requester, in this case a customer as an example (in other scenarios this may be another external entity).

The service requester has a service invocation relationship with the service provider and indirectly with the service

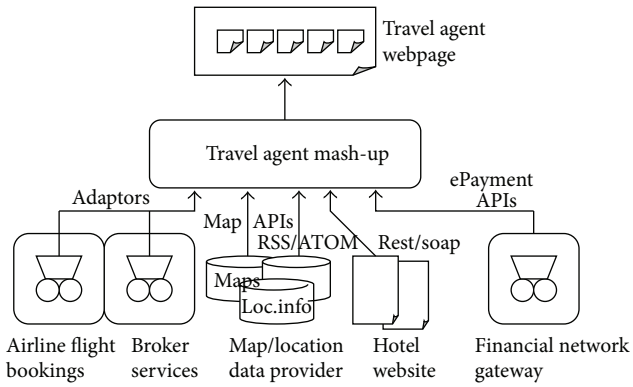

Figure 8: Travel booking mashup application.

source (3rd party provider). In this example, there is no need for additional discovery interaction with the service broker as this is managed by the service provider on behalf of the service requester. Based on these interactions, it is necessary for the service requester to supply correct personal information to reserve a booking and ensure that sufficient funds are available to complete the initial transaction booking. Using the metamodel, a similar analysis is conducted for the remaining entities in this scenario. This process will yield an initial high-level accountability model. The completed analysis is shown in Table 2.

5.1.2. Solution Architecture and Design Stage. The accountability requirements can impose a burden on implementation for the project team. Some key decisions that system

Table 2: Accountability model in travel agent mashup solution.

Entity	Role	Responsibility	Outcome
Customer	Service requester	(i) Provide correct personal information for booking. (ii) Provide sufficient funds upon booking confirmation.	Booking and payment submitted.
Travel agent	Mashup service provider	(i) Provide the overall travel booking functionality. (ii) Conform to service source terms and conditions.	Requests submitted to all service sources. Booking receipt provided to customer.
Mapping provider	Service source	(i) Provide map service. (ii) Adhere to service level agreement.	Map service provided.
Location provider	Service source	(i) Provide location information. (ii) Adhere to service level agreement.	Location service provided.
Airline provider	Service source	(i) Provide ticket booking service. (ii) Adhere to service level agreement	Flight ticket booked and receipt provided.
Hotel provider	Service source	(i) Provide hotel booking service. (ii) Adhere to service level agreement.	Hotel booked and receipt provided.
ePayment provider	Service source	(i) Provide payment service. (ii) Adhere to service level agreement.	Travel agent and service sources receive payment.

Table 3: Accountability requirements.

Interaction scenario	Accountability requirement
Customer ↔ travel agent	(i) Mutual authentication between customer and travel agent. (ii) Disclose traceability in service composition and the up-to-date booking status upon request. (iii) Provide tamper proof evidences on booking transactions upon request.
Travel agent ↔ mapping service Travel agent ↔ location provider Travel agent ↔ fight provider Travel agent ↔ hotel Travel agent ↔ eCommerce	(i) Mutual authentication between travel agent and the external service sources. (ii) Ensure integrity of the service source. This implies fulfillment of the requirements in the terms and conditions of the service source, ensuring the service providers' copyright and trademarks are not breached. (iii) Service source's reputation is tracked and measured against the agreed quality of the services. (iv) Provide tamper proof evidences on service transactions upon request.

designers will make during architecture and design revolve around what to leverage from in-house capability, versus what is provided by external service brokers. We analyze this perspective.

Considering the expanded roles and responsibilities the metamodel (Figure 5) highlights the various service broker roles that provide a different accountability capability. For instance, the travel agent is able to use a rating agent to evaluate and track the reputation of the service source. Alternatively, it may engage an external service dynamically to perform this task, reducing the technical complexity of the solution for the travel. This also implies that it will be necessary for service brokers to provide certain accountability capabilities.

Using the initial high-level roles and responsibility identified in Table 2, we now illustrate the expanded accountability requirements for each interaction scenario (Table 3) to address the issues of trust, integrity, disclosure, and nonrepudiation.

5.1.3. Solution Implementation. In the following, we assume the use of a modeling tool such as the control case [25] to capture and expand the accountability requirements listed in Tables 2 and 3. This may then be machine interpreted during solution implementation.

Broadly speaking, there are two categories of requirements identified in Table 3. The first consists of the obligations that form part of the agreement between the different parties. The second category consists of policies that one entity must comply with in order to consume the service provided by the other party. Examples of the first category of requirements are the terms and conditions of the mapping service source, quality of service (QoS) in service level agreements (SLA), and the travel agent's obligation to provide a booking receipt to the customer upon booking confirming. Together with the ontology, these are used to prepare agreements (see Section 5.2). Conversely, ePayment requirements for an authentication scheme, encryption technologies, and digital signature protocols are examples of the second category of requirements.

```
<wsag:ServiceDescriptionTerm wsag:Name="submitBooking"
      wsag:ServiceName="TravelBooking"/>
      <wsag:GuaranteeTerm wsag:Name="Receipt">
       <wsag:ServiceScope>
           <wsag:ServiceName>TravelBooking</wsag:ServiceName>
       </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>supplied</wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

Algorithm 1: WS-Agreement source.

```
<wsp:Policy       xmlns:wsp=="http://schemas.xmlsoap.org/ws/2004/09/policy"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
      <wsp:ExactlyOne>
       <wsp:All>
            <wsse:SecurityToken>
            <wsse:TokenType>wsse:x509v3</wsse:TokenType>
            </wsse:SecurityToken>
          <wsse:EncryptedParts>
                <wsse:Body />
           </wsse:EncryptedParts>
           <wsse:SignedParts>
                 <wsse:Body />
          </wsse:SignedParts>
       </wsp:All>
      </wsp:ExactlyOne>
</wsp:Policy>
```

Algorithm 2: WS-Policy source.

WS-Agreement [26] may be used to specify the first category of requirements while WS-Policy framework may be used to specify the second category of requirements. These specifications are designed to enable a service provider to advertise the capabilities that form the basis of an agreement. WS-Agreement also enables a requester to negotiate an agreement with the service for a specified duration; furthermore the agreement can be monitored for compliance in runtime [27]. An example of XML source output of the WS-Agreement for the travel agent's obligation of providing booking receipt is shown in Algorithm 1.

The second category of requirements can generally be expressed in a domain specific policy assertion language and may be attached to the service endpoint through the WS-Policy framework. Algorithm 2 is an example of the ePayment requirements for authentication, encryption and digital signature specified using WS-Policy.

5.2. Preparing Agreements for Accountability. We now describe how the ontology may be used to assist in the preparation and negotiation of contracts and agreements between the various entities involved in the travel agent mashup website project. Some key artifacts include the statement of work (SOW) signed between the travel agent and the IT project delivery team and several agreements between the travel agent and external parties acting as a service or content sources. Moreover, the SOW determines the functionality to be provided by the website, the roles, and responsibility for delivering those capabilities and thus sets the basis for all other contracts.

5.2.1. Traditional versus Mashup Agreements. Differing from the traditional IT projects, the mashup project presents extra complexity in the area of accountability. The general accountability ontology defined in Figure 6 and the mashup accountability ontology defined in Figure 7 can assist the project team to prepare an SOW that suitably addresses the accountability issue by identifying accountability elements such as entities, roles, responsibilities, and expected outcome in the service arrangement. One important observation to make is that the ontology outlines the need for accountability tasks associated with each service task. The accountability tasks include disclosing authorities, keeping evidences of the service, and reporting service outcome. In this way, the accountability tasks can be categorized as a form of nonfunctional requirements in IT terms, which can then be reflected in the work breakdown structure, service level agreements, pricing, and the service contract with service source or service broker.

A further observation is that the ontology highlights the multiple roles involved in delivering the web services. This means that an SOW for a mashup solution will differ

```
<profile:serviceParameter>
      <profile:serviceParameterName>
            Type_of_Role
      </profile:serviceParameterName>
      <profile:sParameter>
            Mashup Service Provider
      </profile:sParameter>
      </profile:serviceParameterName>
      <profile:serviceParameterName>
            Invoked Service Source
      </profile:serviceParameterName>
      <profile:sParameter rdf:resource=https://hotel.com/
                                    OnLineBooking/''/>
      </profile:serviceParameterName>
</profile:serviceParameter>
```

Algorithm 3: OWL-S service parameter extension.

from the traditional SOW. In general, the traditional SOW is only required to specify roles and responsibilities of two parties, the project owner and the client (contracts with subcontractors are addressed by additional SOWs). This tenet is supported by Martin, who defines an SOW as "*a narrative description of the products and services to be supplied to the client and the needs and requirements of the contractor to deliver such products and services properly under the contract*" [28]. However, in a mashup case, the SOW definition needs to be broadened to include the roles and responsibilities of all the involved parties including the client, contractor who develops the website, and various service sources. This is because the product and services of the mashup site are not supplied by a single party, rather, by several entities in realtime.

5.2.2. Identifying and Enforcing Accountability. Involved parties are able to apply the ontology to facilitate discovery of accountability statements and clauses. For example, when specifying the hotel booking service within an SOW, based on the ontology in Figure 7, the analyst knows that the hotel operator assumes the web service source role, who performs a service task through a hotel booking web service. A web service has service grounding (information for accessing the service), service process (service logic) and a service profile (service descriptions and service effect). As each service task has associated accountability task, example of accountability statements for the hotel operator could be defined as follows.

(i) Hotel operator is accountable for disclosing the booking service effect and service description metadata that is necessary to locate and access the web service.

(ii) Hotel operator is accountable for providing evidence of the service rendered upon request from the travel agent.

The supplied evidence may be tamper-proof audit trails, service level agreement (SLA) monitoring records, or digitally signed messages. The content and format of the evidence is to be clearly defined in the SOW or in the detailed technical design document referenced by the SOW. Using the ontology in this way prompts the analyst to consider these aspects and include the relevant statements.

Next, the analyst will review the accountability considerations for the travel agent. This is considered in the context of the service provider role, which supplies the travel booking service via the mashup web site. Referring to the ontology, the service provider role is a super type of service source role. In addition to the accountability tasks discussed above, these ontology elements imply that the travel agent is also accountable for disclosing the information in service composition, such as the service profile of all web services provided by the service source roles. The linkage is clearly seen in the ontology as service provider role $\rightarrow$ web service $\rightarrow$ service profile. In the example scenario, the roles include the hotel operator, airline, map service, and location information providers.

The mashup service also requires a valid contract with the consumer. This can be normally achieved by notices and "click-through" agreements when a customer registers with the site. Notices and click-through agreements satisfy the basic requirements necessary to establish legally enforceable commercial transactions [29]. The notices and click-through agreements need to be documented as part of the functionality provided by the website in the SOW.

Examining the consumer role in Figure 7, it is clear that a service requester initiates a service request. Based on the ontology, a service request is a special task. A task delivers certain requirements, which are set out in some form of obligations in a client-supplier agreement that stipulates clearly the terms and conditions of the service provided. To hold the consumer accountable in the travel booking service scenario, the agreement is provided online and must be agreed upon before using the service. Upon accepting the agreement, the consumer will be held accountable for the service request. This may be achieved through the consumer's obligation to provide correct payment details and evidence of payment, such as a valid account, payment card, or eCommerce transaction.

To ensure that accountability is upheld in the mashup services, it is necessary that the SOW stipulate several additional areas. This includes privacy, copyright or trademark laws, and the remaining accountability tasks performed by the relevant parties. Where there is no explicit regulation to be applied, but obligations from involved parties are expected, then a valid contract is required to ensure governance by common law. The ontology assists in this legal analysis by identifying the relevant roles, responsibilities, and service tasks.

6. Summary and Conclusions

As mashup technology enters into the mainstream enterprise business, accountability will emerge as a key requirement to be addressed. As pointed out in [2], a number of legal issues require consideration when using mashup solutions. We suggest that it will be increasingly important for mashup service oriented solutions to have an accountability mechanism builtin to facilitate trust and the resolution of the legal issues.

This paper builds upon existing theories by applying trust and nonrepudiation in a multiparty environment for defining accountability. Using this definition, we propose models that may be used by information systems developers to understand the roles and responsibilities that need to be accommodated in a mashup service solution. In addition, the liabilities and obligations are analyzed. The proposed ontology helps to define the various entities and artefacts involved in a mashup service. This can be used to assist in the preparation of agreements that are required between the various entities involved in a mashup service environment. This will also help to define the scope of accountability to be addressed and will further serve to define requirements of the information systems supporting the mashup service solution in order to meet disclosure requirements.

6.1. Further Work. In addition to developing accountability assertion policies that may be included within the WS-* frameworks, there are further extensions that may be studied. For instance, accountability information such as roles and service composition are currently not available in registry services. In other words, there is no information that advises whether the service being discovered is a mashup service. Nor are there details regarding the various service sources involved. An extension may be developed to supply this additional information to the service requester. This may be useful to assist in deciding if the service being discovered is of sufficient integrity (accountable and accurate) to suite the needs of the requester.

One suggested approach to address this requirement is by including the accountability metadata within the OWL-S service property; specifically one may use the *Service Parameter* property as an extension to add the accountability relevant elements into the service profile. Extending the example from Section 5, the sample XML source shown in Algorithm 3 shows the mashup service utilizing a hotel booking service from "hotel.com."

In addition to machine interpretation of the accountability roles and responsibilities, the proposed ontology may also be used to establish a common accountability vocabulary useful to developers and runtime interpretation. The ontology already contributes to the definition and preparation of agreements between the various parties, and extending the ontology for machine interpretation is an area of further work.

Furthermore, as the mashup services based on RESTful web services are becoming mainstream due to the simplicity and performance benefits that the lightweight architecture offers, inevitably we need to adopt RESTful web service semantic description standards to specify the accountability metadata. Another area of further work is using SAWSDL [30] to annotate accountability metadata for RESTful mashup services.

Acknowledgment

The authors would like to thank Yan Wang for his constructive feedback and helpful comments on thier earlier work on accountability in mashup services.

References

[1] T. O'Reilly, *What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software*, O'Reilly Media, 2005.

[2] R. S. Gerber, "Mixing it up on the web: legal issues arising from internet mashup," *Intellectual Property and Technology Law Journal*, vol. 18, no. 8, 2006.

[3] Performance-Based Management Special Interest Group, *The Performance-Based Management Handbook: Establishing Accountability for Performance*, vol. 3, Oak Ridge Associated Universities, 2001.

[4] J. Zou and C. J. Pavlovski, "Towards accountable enterprise mashup services," in *Proceedings of the IEEE International Conference on e-Business Engineering*, pp. 205–212, Hong Kong, China, October 2007.

[5] D. Martin, M. Burstein, J. Hobbs et al., "OWL-S Semantic Markup for Web Services. W3C Member Submission," 2004, http://www.w3.org/Submission/OWL-S/.

[6] A. Jhingran, "Enterprise information mashups: integrating information, simply," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*, pp. 3–4, Seoul, Korea, 2006.

[7] R. Smith and SOA, *Enterprise Mashup Services. Part 1: Real-World SOA or Web 2.0 Novelties?* SOA World Magazine, 2007.

[8] R. Kailar, "Reasoning about accountability in protocols for electronic commerce," in *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 236–250, May 1995.

[9] R. Kailar, "Accountability in electronic commerce protocols," *IEEE Transactions on Software Engineering*, vol. 22, no. 5, pp. 313–328, 1996.

[10] S. Bhattacharya and R. Paul, "Accountability issues in multihop message communication," in *Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology*, pp. 74–81, Richardson, Tex, USA, March 1999.

[11] M. M. Tseng, J. S. Chuan, and Q. H. Ma, "Accountability centered approach to business process reengineering," in *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 345–354, January 1998.

[12] Y. Zhang, K. J. Lin, and T. Yu, "Accountability in service-oriented architecture: computing with reasoning and reputation," in *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE '06)*, pp. 123–131, Shanghai, China, October 2006.

[13] S. Eriksén, "Designing for accountability," in *Proceedings of the second Nordic conference on Human-computer interaction*, vol. 31 of *ACM International Conference Proceeding Series*, pp. 177–186, Aarhus, Denmark, 2002.

[14] *Webster's Ninth New Collegiate Dictionary*, Merriam-Webster, Chicago, Ill, USA, 1991.

[15] D. G. Johnson and J. M. Mulvey, "Accountability and computer decision systems," *Communications of the ACM*, vol. 38, no. 12, pp. 58–64, 1995.

[16] D. Huang and S. Bracher, "Towards evidence-based trust brokering," in *Proceedings of the 1st International Workshop on Value of Security through Collaboration*, pp. 43–50, September 2005.

[17] S. C. Certo, *Principles of Modern Management: Functions and Systems*, William C. Brown Publishers, Ames, Iowa, USA, 2nd edition, 1983.

[18] B. Frost, *Measuring Performance*, Fairway Press, 1998.

[19] C. Ellison and B. Schneier, "Ten risks of PKI: what you're not being told about public key infrastructure," *Computer Security Journal*, vol. 16, no. 1, pp. 1–7, 2000.

[20] B. Schneier, *Secrets & Lies, Digital Security in a Networked World*, John Wiley & Sons, New York, NY, USA, 2000.

[21] Z. Stojanovic and A. Dahanayake, Eds., *Service-Oriented Software System Engineering: Challenges and Practices*, IGI Global, 2005.

[22] K. J. Ma, "Web services: what's real and what's not?" *IEEE IT Professional*, vol. 7, no. 2, pp. 14–21, 2005.

[23] J. Luo, B. Montrose, A. Kim, A. Khashnobish, and M. Kang, "Adding OWL-S support to the existing UDDI infrastructure," in *Proceedings of the IEEE International Conference on Web Services (ICWS '06)*, pp. 153–160, Chicago, Ill, USA, September 2006.

[24] D. Martin, M. Burstein, O. Lassila, M. Paolucci, T. Payne, and S. McIlraith, "Describing web services using OWL-S and WSDL," DAML-S Coalition working document, 2003.

[25] Z. Joe and C. J. Pavlovski, "Modeling architectural non functional requirements: from use case to control case," in *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE '06)*, pp. 315–322, Shanghai, China, October 2006.

[26] A. Andrieux, K. Czajkowski, K. Keahey et al., "Web services agreement specification (WS-Agreement)," Grid Resource Allocation Agreement Protocol (GRAAP) Working Group, Open Grid Forum, 2005.

[27] B. Margolis and J. Sharpe, *SOA For the Business Developer: Concepts, BPEL, and SCA*, MC Press, 1st edition, 2007.

[28] M. Martin, "Statement of work: the foundation for delivering successful service projects," *PM Network*, vol. 12, no. 10, pp. 54–57, 1998.

[29] J. Matsuura, *Security, Rights, and Liabilities in E-Commerce*, Artech House, Norwood, Mass, USA, 2002.

[30] W3C, "Semantic annotations for WSDL and XML Schema," 2007, http://www.w3.org/TR/sawsdl.