Fox, M.S., (2012), "Managing eCommerce Service Failures: Incorporating Validity, Provenance and Trust from the Semantic Web." Proceedings of the 14th Annual International Conference on Electronic Commerce (ICEC14), Singapore.

Managing eCommerce Service Failures: Incorporating Validity, Provenance and Trust from the Semantic Web

Mark S. Fox
Department of Industrial and
Mechanical Engineering
University of Toronto
5 King's College Rd, Toronto ON
M5S 3G8 Canada
+1-416-978-6823 msf@eil.utoronto.ca

ABSTRACT

High volume, B2C eCommerce web sites that often experience millions of visitors and 100,000s of orders a day in the lead up to major gift giving occasions, must meet customer demands of high availability, responsiveness, and functionality. These same web sites often integrate a large number of 3rd party web services that need to but often fail to meet the same demands. This paper describes a Semantic Web-based solution to meeting such demands. By distinguishing between action vs information, required versus optional, time constrained, and substitutable services, combined with the semantic web concepts of information validity, provenance and trust, more targeted and flexible responses to service failures can be provided.

Categories and Subject Descriptors

H.3.5 Online Information Services. I.2.4 Knowledge representation Formalisms and Methods;

General Terms

Web Services, Semantic Web, Ontology.

Keywords

Semantic Web, Information Provenance, Information Validity, Trust, Web Service Failure, Ontology. OWL, RDF, XML

1. INTRODUCTION

B2C eCommerce, and in particular, online retailing (eRetail) presents an interesting set of challenges for those sites that rely upon web services, whether they be internal or 3rd party. Consider a gifting web site such as flowers. Customer volumes and demands on web services can be easily planned, for "between occasions" periods of time. But the lead up to gift giving occasions, such as Mothers Day, Christmas, Duvali, Chinese New Year, etc. see visitor and order volumes grow by 2 orders of magnitude into the millions of visitors and 100,000s of orders. Regardless of the volumes, customers expect the same user experience: high availability, fast response times, 24x7, and no reduction in services, such as promised delivery dates, inventory availability, coupon processing, payment processing, etc.

Unbeknownst to most people, today's eRetail systems integrate many 3rd party services in order to deliver their user experience. These include: payment processing, product recommendations, customer product reviews, taxation, shipping pricing, address verification, and fraud analysis. These services have to satisfy the same customer expectations. Yet, service failures occur regardless of best laid plans. Often, these failures occur at the time of highest customer demand, as that is when boundary conditions within the services are stressed.

Much work has gone into the development of general methods for responding to service failures. This paper proposes that another layer of representation and reasoning needs to be defined on top of the general methods in order to tailor the response to service failures to meet the special needs of B2C eCommerce:

- Some services are critical to the operation of the web site. For example, payment processing services are critical as they determine whether a credit card is valid and whether the purchaser has enough "room" available on their card to make the purchase. Order processing cannot proceed without credit card approval.
- Some services are strictly informational. For example, product reviews are not critical. The frequency of update can be low without affecting the operation of the web site. On the other hand, foreign exchange rates are much more important and need to be updated more frequently. Never the less, while these services are not critical, like payment processing, they do affect Customer Order conversion rates and Average Order Value, which in turn affects profitability[1].

It is clearly the case that not all services are created equal; that some services are more important that others. Some are necessary for the operation of the web site, some are not. Some require frequent updates, some don't. Our goal is to look more closely at web services in the context of B2C eCommerce to see how their characteristics lead to more specific methods of handling service failures.

In the following we provide two examples of eRetail web sites and analyse the types of web service they utilize and how critical they are to their operation. We then review some of the relevant literature in web service failure management and semantic web representations of validity, provenance and trust. Next we list the key requirements that a solution to managing service failure satisfy followed by our approach to solving the problem. Finally we provide a short example of how information services are represented using the semantic web in order to support failure management.

---

[1] eRetailers continuously improve their user experience to boost revenues. Revenue gains of 5% are significant. Good recommendations, reviews, etc. can often boost revenues by this much or more.

2. MOTIVATION

Riverbed.ex (not their real name) is an online seller of household products and gifts, including consumer electronics, lifestyle products, games, and toys. Their annual online sales exceeds $100 million with an average order value of $200. 60% of their sales occur during the last 3 months of the year. During that period they process about 300,000 orders, peaking at over 10,000 orders/day in the run up to Christmas. Their web site integrates with over 15 external web services covering payment processing, product reviews and recommendations, shipping, email marketing, taxes and click stream analysis.

Critical to their operation is payment processing because without it orders would have to be queued for manual processing by their call centre which is a very expensive process. Alternate payment methods, such as Paypal are important but can the site can operate without it as long as the primary payment processor of credit cards is operational. Reviews and recommendations are also very important as they can increase Average Order Value significantly. The site can operate without these third party services but at a lower level of revenue and profitability. Tax and shipping information is important, but shipping and tax tables can be cached and if there is a discrepancy between the cached versus actual version, the eRetailer "eats the difference". Finally, inventory information, if not available, the latest version can be cached but may result in orders being accepted when the product is out of stock[2] .

Flowersrus.ex (not their real name) is an online seller of floral arrangements that aggregates and distributes orders to a network of floral shops for fulfillment. Their annual online sales exceeds $400 million with an average order value of $60. On the peak day in the run up to Valentines Day and Mothers Day, they process over 100,000 orders. They process over 100 orders/second during peak periods. Their web site integrates with a number of internal and external web services, including payment processing, inventory availability, address verification,

Critical to their operation are payment processing and availability of floral shops to fulfill. Regarding the latter, if there are over 10,000 floral shops that have to fulfill from a catalog of 1,000 products, then there is 10,000,000 data points of availability to fulfill where each data point is updated at random times by the floral shops.

Over the years customers have come to expect eRetailers' web sites will be reliable. Reliability has at least three aspects:

1. **Availability**: the web site is always available, especially during critical shopping periods. It cannot shut down when you have 100,000s of customers accessing you site in a day
2. **Responsiveness**: The response time of the site should be under 3 seconds, with the ideal being less than a second.
3. **Functional**: All functional capabilities of the web site, e.g., payment processing, recommendations, coupons, etc., should be available and operating correctly.

Given the number of 3rd party services incorporated in eRetail web sites, there is no guarantee that these services will meet the level of reliability demanded by consumers. Hence, eRetail web sites have to figure out a way to do without services when they become unavailable.

As discussed in the next section, there are general mechanisms that have been proposed to handle web service failures. But their generality limits their usefulness (i.e., ease of use, clarity in how to address eRetail problems). We believe that by developing a deeper understanding of the nature of web service failures in eRetail, more tailored and relevant solutions can be developed.

An analysis of eRetail services leads to the following categorization of the web services that they use:

- **Action or Information:** Does the service perform an action (e.g., payment processing) or provide information (e.g., product review)?
- **Time Constrained or not:** Does the action have to be performed or the information returned/updated within a set period of time?
- **Required or Optional:** Is the action or information required (e.g., Taxes, Shipping charges) for the successful operation of the web site or can it be omitted (e.g., product reviews)?
- **Substitutable or not:** Do there exist other services that can be substituted for a failed service (e.g., alternative payment processors).

Depending on the category of a web service, the response of the web site may vary from seeking alternatives (e.g., payment processors) to completing removing a service from the customer's experience on the web site (e.g., generating recommendations, identifying best sellers). In the situation where an action, such as payment processing is time constrained and required, the alternative of last resort is to accept the order and process it manually.

An additional consideration has to be taken into account, namely data protection. In the case of credit card processing, we cannot rely on the external service to provide the recovery mechanism as they should not be able to possess credit card information for any extended period of time.

Finally, given the customers' demand for reliability, there is one overarching goal that any solution must satisfy: *Service failures*

[2] Multi-channel retailers have their stock drawn down simultaneously by the web site, brick and mortar stores and the call centre. Therefore, the web site has to request stock updates from the ERP system before processing an order. If the connection to the ERP system fails, stock levels stored by the web site may become incorrect.

should not affect the customer experience[3]. To achieve this, the web site should dynamically adapt to service failures thereby maintaining a good customer experience.

3. BACKGROUND

Our approach builds on the growing foundation of the Semantic Web by defining new classes in OWL. OWL is a Semantic Web implementation of Description Logic (OWG, 2009). It extends the RDF/RDFS layer of the Semantic Web with a richer set of XML elements for representing classes and their properties. OWL-S is a Web Service Ontology: a set of OWL classes and properties that represent services, including their input & output, and their process model (Martin et al., 2008). The following summarizes research that has explored how to manage service failures using OWL-S:

- Vaculin & Sycara (2007) introduce a set of application independent error event types and suggest additional OWL-S Process Model Result classes for recognizing application specification errors. These provide their OWL-S Virtual Machine (OVM) with the ability to recognize when service results are in error.
- Vaculin et al. (2008) extend this work with the introduction of constraint violation handlers that define violations using event expressions (i.e., patterns of events). An important event expression, from the perspective this paper, is the ability to specify time limits on processes. The handler specifies how the OVM is to deal with the constraint violation. Possible responses include compensatory actions.

Given the generality of the OWL-S specification, and more specifically the Process Modelling classes, a process model can be defined to perform any type of conditional processing based on the results of a constituent service. Secondly the extensions defined by Vaculin et al. (2008) provide a powerful mechanism for handling failures predefined as event patterns. Never the less, the intent here (as in the work of Vaculin et al.) is to provide additional ontological primitives that make the specification and handling of service errors simpler[4].

Over the last decade, concerns around information validity, provenance and trust have grown. With the web now containing billions of documents authored by millions of people, the need to know whether the content is true, where the content came from and whether to trust its creator has taken on an increasing importance.

Much of the research into provenance has grown out of workflow management where the focus has been the evolution of a document has it proceeds through a sequence of edits, perhaps by many different people. Tracking the various versions created, who did what and when has been the primary concern. This research has culminated in the proposed Semantic Web standard called the PROV model (Belhajjame et al., 2012), which has built on the work of Hartig & Zhao (2010) and Moreau et al. (2010).

Another approach to provenance has employed logic, where web content is viewed as propositions with assigned truth values (Fox & Huang, 2005b) so that the validity of information can be represented. The truth value can change over time allowing for non-monotonicity in what we know (Huang & Fox, 2004a) and can also be uncertain (Huang & Fox, 2004b). This work also supports a provenance model of who created and transformed the information (Fox & Huang, 2005a).

Finally, trust in the creator of information begins with the ability to certify that the creator is who they say they are. This builds on the existing Public Key Infrastructure. But determining whether to trust the information creator turns out to be highly contextual where the decision to trust someone may be based on their reputation, contextual information such as meta-data available on the web about the creator, or other content-based rules (e.g., don't trust weather predictions that are more then 10 days in advance). Examples of the various directions in semantic web-based trust include Goldbeck et al. (2003), O'Hara et al. (2004), Goldbeck (2006), Huang & Fox (2006) and Huang (2008).

4. KEY REQUIREMENTS

The key driver behind the design of an B2C eCommerce web site is not to lose a customer nor a sale. Hence it must satisfy the availability, responsiveness and functional requirements described in section 2. To do so, it must have the ability to:

1. **recognize** when a service has failed,
2. **reuse** prior information, if possible, if it is an information service that failed,
3. **remove** a service from the user experience on the web site when it is optional,
4. **substitute** a service when the service is necessary, and
5. **delay** a service, with customer agreement, if there is no alternative.

5. SOLUTION

Our solution builds upon the exception handling mechanisms introduced in (Vaculin, Wiesner & Sycara, 2008) by assuming there exists a client side implementation of their OVM that provides the basic web service infrastructure and low level constraint failure handling. It is implemented as Java classes which in turn access internal and external web services. Consequently, much of the complexity of working directly with OWL-S and extensions are hidden from the user, thereby simplifying eRetail development.

5.1 Action Service Failure

The key factor in managing action service failures is determining whether a service is required. If required, the web site has to dynamically modify its user experience. For example, if the web site has multiple payment options, and Paypal is not available, it can simply remove Paypal as an option in the checkout process. But if the only payment process service has failed, then it may have to communicate to the customer that payment processing is not available, that it will process the order but put it on hold until it can process the payment. In either case, the web site has to be able to access information about the service, i.e., whether it is required or optional, and whether the service is available, in order to determine how to dynamically modify the user experience.

Determining whether a service is available relies on two time constraints. The first time constraint defines the amount of time the web site is willing to wait for a response from the external

[3] By Customer Experience, we mean the interactions the customer has with the web site.

[4] Related to this have been studies of managing exceptions in B2B/Supply Chain Management (Lin & Chang, 2005). Though interesting, the response time requirements of eRetail, which is measured in seconds, versus days or weeks as in the supply chain, limit its relevance.

service before it declares that the service is no longer available. The second time constraint defines how long the web site is willing to keep trying the service request. In the following, we define the parameters for the Action class used to process an action service failure. We assume that the action service provides credit card processing that is of critical importance to the operation of the eRetail web site.

- A specification of the primary service. For example, a credit card payment processor.
- A specification of one or more substitutable services to be used when the primary is not available.
- A specification of the "last resort" alternative to be used if all others fail.
- A flag that indicates whether the service is required or optional.
- A status flag that is set to false when the service is not available. This allows for checking by the web site that may result in the service being removed from the web site. For example, the web site may offer an alternative billing method, such as Paypal, but can dynamically remove the Paypal from the web site if it is not available.

For each service identified above, the following instance of the Action Service class/object:

- A time limit on how long to wait for a response.
- A time limit on how long to keep retrying the service request. If a service is not available it will cache the request for later processing.
- A cache for failed service requests.
- A status flag that is set to false when this specific service is not available.

The Action class will always attempt to use the primary service. If the primary service fails (i.e., the retry time limit is exceeded), it will attempt each substitutable service in sequence. For each attempt, it will transfer the cached requests to the current service. At each step of the process, it will recheck the prior services to see if they are again available. Each Action Service class will continually check the availability of its corresponding service and the change its status flag accordingly.

There has been no attempt here to utilize dynamic service discovery as a means of finding substitutable services. For critical services, such as payment processing, the contract negotiation process remains manual and lengthy. Therefore, contracts need to be in place with service providers well in advance.

5.2 Information Service Failure

Previous work on web service failures treats information provisioning like any other service. Our approach is to look more closely at information services and to see how their characteristics lead to more specific methods of handling service failures.

Consider an external service that provides product reviews. A review is composed of a product identifier, rating, review, and reviewer identifier. There are a number of failure modes that may exist:

- The service may become unavailable for an extended period of time,
- The service may find that the review contains information that it no longer wishes to be available, or
- The reviewer is found to be untrustworthy and the review should be removed altogether.

In the case of service failure, the product review service is no longer operational due to application, server or network failures on the service side. This can be easily recognized on the client side using temporal constraints. A simple mechanism for dealing with the loss of an information service is to cache the information and continue to use the cache until the service becomes available. But simple caching does not address the problem of information criticality. One dimension of information criticality is its temporal validity; how long can the information be used before it should be discarded. For reviews, the temporal validity can be measured in days if not weeks, but for foreign exchange rates the temporal validity may be measured in minutes. Temporal validity of information defines how long from the last update before it is no longer valid and must be discarded. Assuming that the information is provided using Semantic Web standards such as RDF and OWL, we can extend the representation of reviews with temporal validity elements[5]. The temporal elements would provide two date/times: the date/time the information can begin to be used, and the date/time the information is no longer to be used. To specify the temporal validity of information we incorporate our previous work on Dynamic Knowledge Provenance (Huang & Fox, 2004; Fox & Huang, 2005a). DKP (Dynamic Knowledge Provenance) provides the elements for specifying the temporal period during which the information is valid.

In the case where the service provider determines that a review contains information that it no longer wishes to be used by its clients, possibly due to inaccuracies or other errors, we need a means of removing it from the clients' web sites. A simple solution would be the remove the information in the next update retrieved from the service provider by its clients. But if the review has an extended temporal validity, it may be a long time before clients retrieve an update. If a review makes false claims about a product and the manufacturer is threatening to sue, the speed with which the information is removed becomes important (in fact the time of removal may be mandated the courts). Our approach builds on the Semantic Web by providing elements for static information validity. In our work on Static Knowledge Provenance (Fox & Huang, 2005b), we provide the elements for specifying who created the information (i.e., provenance) and the status of its validity, independent of time. Our implementation uses these elements to annotate reviews. Independent of temporal validity, using the provenance and validity information provided, the client can check with the information creator to dynamically determine if the information is still valid. For information whose temporally validity may be long, the client may periodically verify that the information is still valid. Note that this is independent of temporal validity as the information may be temporally valid and the service still available.

In the case where the reviewer is found to be untrustworthy, we need a mechanism that will allow us to:

- Identify who the reviewer is, and
- Determine the trustworthiness of the reviewer

The determination of whether to trust a review can be made by either the client or the service. Representations and reasoning about trust have been explored for some time (Blaze et al., 1996; Finin & Joshi, 2002; Golbeck et al., 2003; O'Hara et al., 2004;

[5] We use the term "element" since within the semantic web the ontological terms will be represented as XML elements.

Golbeck, 2006; Huang & Fox, 2006). Our approach applies the provenance and trust representations of Fox & Huang (2004) and Huang & Fox (2006), respectively, to annotate the review information with elements covering who the originator of the information is, the path it has taken from one URI to another leading to its current location, and context in which the originator can be trusted.

6. IMPLEMENTATION

In this section, examples of the Information Service Failure solution are provided. Classes have been defined using OWL 2 (OWG, 2009) and are serialized using the XML/RDF.

In our example, we start with the specification of a product review. The review ontology is based on the ICECAT Open Catalog specification (www.icecat.biz/get_attachment.cgi?5508).

Given the following RDF namespace declarations for the subsequent examples:

```
<rdf:RDF xmlns:rdf="http://w3.org/TR/1999/02/22-rdf-syntax-ns#"
    xmlns:mie= "http://www.mie.utoronto.ca/faculty#"
    xmlns:rev=
        "http://www.eil.utoronto.ca/ontology/reviews#"
    xmlns:kp="http://www.eil.utoronto.ca/ontology/kp#"
    xmlns:tr="http://www.reviewservice.ex/reviewers#"
    xmlns:cnet="http://www.cnet.com/products#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/ "  />
```

Following is an example of what a review would look like in http://www.reviewservice.com/reviews:

```
<rev:Review rdf:ID="r34567">
    <rev:reviewer rdf:resource="&tr;JoeSmith />
    <rev:product rdf:resource="&cnet;472930 />
    <rev:productCategory
        rdf:resource="&cnet;ConsumerElectronics" />
    <rev:rating>1</rev:rating>
    <rev:mainReview
        This product does not work at advertised.
    </rev:mainReview>
    <rev:whatsGood>Nothing </rev:whatsGood>
    <rev:whatsBad>Everything!</rev:whatsBad>
    <kp:effectiveFrom>2012-01-01</kp:effectiveFrom>
    <kp:effectiveTo>2012-01-31</kp:effectiveTo>
    <kp:assignedTruthValue
                    rdf:Resource="&rev;rr34567tv" />
</rev:Review>
```

The above defines:

- who the reviewer is (rev:reviewer)
- the product being reviewed (rev:product)
- the product category (rev:productCategory) taken from CNET's product categorization
- a rating of the product (rev:rating)
- a main review (rev:mainReview)
- "whats good" about the product (rev:whatsGood)
- "whats bad" about the product (rev:whatsBad)
- a start date for which the review is valid (kp:effectiveFrom). The web site should not display the review prior to its effectiveFrom date.
- an end date at which the review is no longer valid (kp:effectiveTo). If the service provider is not available, the client will continue to use the review until the effectiveTo date is reached. At that point, the review should be removed from the web site.
- the truth value (i.e., validity) of the review to which the review is believed to be true (kp:assignedTruthValue). This link is used to verify that the view is still valid.

The effectiveFrom and effectiveTo dates address the problem of when to start and stop displaying the review if the review service has failed. The review can continue to be displayed until the effectiveTo date. For the case where the review service has decided that the review content should no longer be displayed, they would set the assginedTruthValue to false. Even though it may be within the effective time period, the web site would periodically retrieve the truth value and determine if it is to be removed.

Given http://www.reviewservice.com/reviewers contains the following information about the reviewer Joe Smith:

```
<tr:reviewer rdf:ID="JoeSmith">
   <foaf:name>Joe Smith</foaf:name>
      <tr:hasReview rdf:resource="r34567rev" />
</tr:reviewer>
```

The above simply defines a reviewer Joe Smith (we have left out other information about the reviewer) and that they have one review that they have created. Next we define the trust object that links the review service, the reviewer and the review:

```
<kp:trust_b rdf:ID="t1">
   <kp:trustor rdf:resource=rev:reviewservice/>
   <kp:trustee rdf:resource="&tr;JoeSmith" />
   <kp:trustObject rdf:resource="r34567rev" />
   <kp:truthValue>False</kp:truthValue>
</kp:trust_b>
```

The above states that the review service does not believe in the review produced by Joe Smith. The consequence of this negative belief is that the web site should remove it. While the effect is the same as setting the truth value of the review to False as we did earlier, the semantics is different. By setting the truth value of a review to False, we are saying that the review is no longer valid, for whatever reason. By stating we no longer believe in the review produced by the reviewer, we are making a statement both

about the reviewer and the review. A consequent being we may no longer trust Joe Smith in regards to any other reviews he provides.

7. CONCLUSION

In the world of eRetail, failures are unacceptable. Customers can switch their loyalty at a click of the mouse. The goal of this work is to provide a layer of abstraction between the eRetail application and the underlying service management mechanisms. (Existing web service ontologies such as OWL-S and their failure management extensions provide powerful mechanisms for recognizing and managing service failures.) This abstraction provides the tools necessary for the web site to flexibly respond to service failures in a way that minimizes the impact on the user experience. Central to the approach is the adoption of semantic web standards and the incorporation of ontologies for information validity, provenance and trust.

8. ACKNOWLEDGMENTS

This research was sponsored, in part, by the Natural Science and Engineering Research Council of Canada, and Novator Systems Ltd. where Dr. Fox was Chairman and CEO from 1993 to 2011.

9. REFERENCES

Blaze, M., Feigenbaum, and Lacy, J., (1996), "Decentralized Trust Management", *Proceedings of the IEEE Conference on Privacy and Security.*

Belhajjame, K., Deus, H., Garijo, D., Klyne, G., Missier, P., Soiland-Reyes, S., and Zednik,S., (2012), "PROV Model Primer", http://www.w3.org/TR/2012/WD-prov-primer-20120110/

Finin, T., and Joshi, A., (2002), "Agents, Trust, and Information Access on the Semantic Web", *SIGMOD Record*, Vol. 31, No. 4.

Fox, M.S., and Huang, J., (2005a), "Knowledge Provenance in Enterprise Information", *International Journal of Production Research*, Vol. 43, No. 20., pp. 4471-4492. http://www.eil.utoronto.ca/km/papers/fox-ijpr05.pdf

Fox, M.S., and Huang, J., (2005b), "An Ontology for Static Knowledge Provenance", In *Knowledge Sharing in the Integrated Enterprise,* IFIP, Vol. 183/2005, pp. 203-213.

J. Golbeck, "Combining Provenance with Trust in Social Networks for Semantic Web Content Filtering," in *Provenance and Annotation of Data, International Provenance,* Chicago, IL, USA, 2006

Golbeck, J., Parsia, B., and Hendler, J., (2003), "Trust Networks on the Semantic Web", Cooperative Information Agents VII, Vol. 1, No. 1, Springer, pp. 238-249.

Hartig, O., and Zhao, J., (2010), "Publishing and Consuming Provenance Metadata on the Web of Linked Data", *Proceedings of the Third International Provenance and Annotation Workshop*.

Huang, J., (2008). "Knowledge Provenance: An Approach to Modeling and Maintaining The Evolution and Validity of Knowledge", PhD Thesis, Dept. of Mechanical and Industrial Engineering, University of Toronto. http://www.eil.utoronto.ca/km/papers/huang-PHD-2007.pdf

Huang, J., and Fox, M.S., (2004a). "Dynamic Knowledge Provenance", *Proceedings of Business Agents and Semantic Web Workshop*, pp. 372-387, National Research Council of Canada, London Canada. http://www.eil.utoronto.ca/km/papers/huang-nrc04.pdf

Huang, J., and Fox, M.S., (2004b), "Uncertainty in Knowledge Provenance", *Proceedings of the European Semantic Web Symposium*, Springer Lecture Notes in Computer Science. http://www.eil.utoronto.ca/km/papers/EuroSemWeb04-online.pdf

Huang, J., and Fox, M.S, (2006), "An Ontology of Trust – Formal Semantics and Transitivity," *Proceedings of the International Conference on Electronic Commerce,* pp. 259-270. http://www.eil.utoronto.ca/km/papers/huang-ec06.pdf

Lin, F., and Chang, H., (2005), "B2B E-Commerce and Enterprise Integration: The development and evaluation of Exception Handling Mechanisms for Order Fulfillment Process basedon BPEL4WS", *Proceedings of the 7th International Conference on Electronic Commerce*, ACM, pp. 478-484.

Martin et al., (2008) "OWL-S: Semantic Markup for Web Services", http://www.ai.sri.com/daml/services/owl-s/1.2/overview.

Moreau et al., (2010), "The Open Provenance Model Core Specification (v1.1)", *Future Generation Computer Systems*. Also see openprovenance.org.

O'Hara, K., Alani, H., Kalfoglou, Y., and Shadbolt, N., (2004), "Trust Strategies for the Semantic Web," in *Proceedings of the ISWC*04 Workshop on Trust, Security, and Reputation on the Semantic Web,* Hiroshima, Japan.

OWG: OWL Working Group, (2009), "OWL 2 Web Ontology Language", http://www.w3.org/TR/owl2-overview/

Vaculin, R., and Sycara, K., (2007a), "Monitoring Execution of OWL-S Web Services", *Proceedings of the European Semantic Web Conference: OWL-S Experience and Directions Workshop*.

Vaculin, R., and Sycara, K., (2007b), "Specifying and Monitoring Composite Events for Semantic Web Services", *Proceedings of the IEEE European Conference on Web Services,* IEEE Computer Society, pp. 87-96.

Vaculin, R., Wiesner, K., and Sykara, K., (2008), "Exception Handling and Recovery of Semantic Web Services", *Proceedings of the Fourth International Conference on Networking and Services*, IEEE.