

Selective Querying using Value of Changed Information for Adapting Hierarchical Web Service Compositions

John Harney
LSDIS Lab, Dept. of Computer
Science, University of Georgia
Athens, GA 30602
jfharney@uga.edu

Prashant Doshi
LSDIS Lab, Dept. of Computer
Science, University of Georgia
Athens, GA 30602
pdoshi@cs.uga.edu

ABSTRACT

Environments in which Web service compositions (WSC) operate are often dynamic. We address the problem of which service to query for up-to-date information in order to adapt a hierarchical WSC, given that queries are not free. Previously, the value of changed information (VOC) has been proposed to select those services for querying whose revised non-functional information is expected to bring about the most change in the composition. VOC requires a distribution over the possible values of volatile parameters of the WSs. In this paper, we present an approach for utilizing VOC in the context of a WSC composed of WSs and lower level WSCs, which induces a natural *hierarchy* over the composition. Because parameters of composite WSs are not directly available, we aggregate these from parameters of component WSs and derive a distribution over the possible parameter values from the distributions for the WSs.

Categories and Subject Descriptors

H.3.5 [Information Storage and Retrieval]: Online Information Services – Web-based Services; G.3 [Probability and Statistics]: Probability density functions

General Terms

Theory, Algorithms

Keywords

hierarchy, Web service composition, revised information

1. INTRODUCTION

Approaches for composing Web services assume that the parameters used to model the environment remain static and accurate throughout the composition's execution. This fundamental assumption is unrealistic as environments are subject to change during execution. For example, a product may go out of stock affecting the availability, the network bandwidth may fluctuate affecting the WS response time, or the cost of invoking a travel agent's service may increase. Many Web service composition (WSC) techniques do not adapt compositions to such changes, leading to suboptimal results.

Dynamism manifests in WSC environments in a variety of ways. For example, changes range from the operational level

Copyright is held by the author/owner(s).

WWW2009, April 20-24, 2009, Madrid, Spain.

(such as a newly introduced task) to the organizational level (such as new company policies) as mentioned in [13, 20]. Indeed, these surveys classify a variety of changes in different ways. Solutions have been presented to address some of these changes, ranging from exception handling techniques defined in [3] to instituting protocol adaptations in [8].

However, less attention has been paid to *data volatility* that exists during execution. As a concrete example, consider a mortgage loan acquisition process in which two title insurance agencies compete for orders from a large mortgage broker. The sequence in which the broker utilizes the services of the two insurers would depend on the probability with which the insurers usually satisfy the requests and the cost of using them. If the preferred insurer's rate of request satisfaction drops suddenly (due to say, a financial crisis), a cost-conscious broker should replace it with another insurer to remain optimal. Important non-functional service parameters such as cost, availability, or the rate of request satisfaction as in the above example, often change during the life-cycle of a WSC. WSCs must be aware of the changing parameters of the participating services so as to optimize the composition.¹

Thus, the WSC must possess up-to-date knowledge of the revised information during execution. To obtain this knowledge, an adaptive WSC may query services – typically their providers – for the services' revised parameters. The revised values are then integrated into the model so that the composition is optimal.

Querying for component services' parameters, however, comes with its own attendant challenges. While revised information about some services may cause changes in the overall WSC, changes to other services' parameters may have little or no impact on the WSC. Additionally, WSCs typically operate over an open and large scale system (the Web). As a result, querying for information from service resources is often tedious, time consuming, and costly. Queries must therefore be carefully managed – we should query only those services whose parameter changes may potentially impact the WSC so as to minimize any overhead introduced. Specifically, the adaptive WSC should know: (1) when is it cost effective to query for the changed information and, (2) which service(s) to query.

In [14, 15], Harney and Doshi introduced a method to intelligently query for revised parameters using the *value of changed information (VOC)*. In particular, the approach computes the trade-off between the cost of querying for up-

¹For the sake of simplicity, we assume that changes in the composition do not upset the consistency of the WSC.

to-date information and the value of expected change in the WSC that the revised information will bring. The model parameters are updated and the WSC is composed again, only if the VOC is greater than the query cost. In computing the VOC, the approach utilizes stochastic models of volatility of each of the services’ parameters. The approach is *myopic* in that only one service provider is queried at a time, and the revised information for that WS is utilized which leads to the maximum VOC.

While previously the usefulness of VOC was demonstrated in the context of simple WSCs [14], in this paper, we expand the applicability of VOC to a *hierarchical* WSC. In many cases, to promote scalability a WSC may be seen as nested – a higher level WSC may be composed of WSs and lower level WSCs – which induces a natural hierarchy over the composition. In contrast to flat WSCs, a hierarchical decomposition introduces multiple challenges: (a) Because only the parameters of the lower level component services are known, we must derive the parameters of the composite service from these, and (b) we must derive a model of volatility for the composite service’s parameters from the models of its component services. While approaches for aggregating parameters of component WSs exist [4, 23], we present a way of formulating the stochastic models of volatility of composite services. Given approaches that address both these challenges, we show how we may compose and adapt hierarchical WSCs by descending down the levels of nesting and computing the VOC for WSCs at each level. The procedure is further complicated if we decide to query a composite service.

We show that, though myopic, our approach performs well in the presence of data volatility exhibiting reduced costs in comparison to the approach of not adapting and other heuristical ways of querying. In particular, our experiments demonstrate that the VOC mechanism avoids “unnecessary” queries in comparison to naive approaches of querying, say periodically. While it is impossible to predict queries that will result in revised information which is guaranteed to change the WSC, we empirically demonstrate that considering VOC significantly reduces false positive queries – those which do not result in a changed WSC. This translates to savings in overall costs for the WSC. For the purpose of evaluation, we utilize a realistic scenario of mortgage loan processing. Within our services-oriented architecture (SOA), we represent the mortgage broker’s WSCs using WS-BPEL, and the different provider services as well as a service for computing the VOC using WSDL.

2. MOTIVATING SCENARIO: MORTGAGE LOAN PROCESSING

Our scenario is a simplified version of a mortgage loan acquisition process, typically used by broker businesses that service mortgage loans to individual clients. The broker uses a WSC to automatically process a mortgage loan request. The composition engages a variety of external WSs and vendors to obtain information crucial to securing a suitable home mortgage loan for a client. The broker aims to ensure a consistent and robust workflow while maintaining minimal expenditures.

Figure 1 illustrates an example process utilized by the broker. We begin with a description of the upper level of the hierarchy (labeled “Level 2”). The broker receives a request

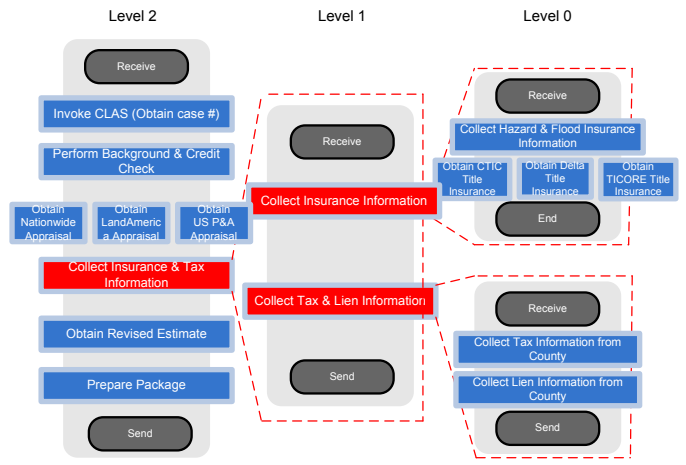


Figure 1: A hierarchical mortgage loan process used by a mortgage broker. The broker may choose among multiple appraisal companies and title insurance companies at different levels in the composition.

from a client interested in securing a mortgage loan. Some of the activities that the broker performs are to issue a request to the CLAS (CHUMS Lender Access System) WS to obtain a case number for this particular client, utilize the WS of the credit rating agencies to perform financial background and credit checks of the client, and hire an appraisal agency to appraise the value of the home. In this example, the broker may choose between three different appraisal services: Nationwide, LandAmerica, and US P and A. A choice among these is made depending on the cost and the availability of using the service at a given time. If a chosen service is unable to perform the appraisal, then one of the other appraisal services may be used instead. Next, the broker invokes the insurance and tax information collection service to obtain vital home insurance, tax and lien information. With all of this information on hand, the broker will complete an estimate of the loan and prepare the complete package for the client to review.

We formulate the **Insurance and Tax Information** service (shown in red) as a composite service. The corresponding lower level (level 1) composition consists of two services for collecting insurance information and collecting tax and lien information. These services are themselves composite. The **Insurance Information** service involves gathering information on both hazard and title insurances. Note that at this stage, the broker must decide on a vendor (CTIC, Delta, or TICORE) to provide the title insurance. Finally, the **Tax and Lien Information** service invokes home county services to extract the tax and lien information on the home. The costs incurred to the broker hinge, in part, on the probability with which the appraisal services and the title insurance services may process a request. For example, if the request completion rate of a previously chosen appraisal service (such as Nationwide) suddenly drops, the WSC must adapt (ie. utilize a different service) to remain cost effective.

Therefore, the example reveals two important factors for choosing optimally. First, the broker must know the certainty with which the appraisal requests will be satisfied by each of the agencies. Second, at each stage, rather than

greedily selecting an action with the least cost, the broker must select the action which is expected to be optimal over the long term.

3. BACKGROUND

For the purpose of illustration, we select a decision-theoretic planning technique for composing WSCs [9]. However, our approach is applicable to any model-based service composition technique such as [1] (for example, see [6]).

3.1 Web Service Composition Using MDP

Decision-theoretic planners such as MDPs model the composition environment, WP , using a sextuplet:

$$WP = (S, A, T, C, H, s_0)$$

where $S = \prod_{i=1}^n X^i$, S is the set of all possible states factored into a set, X , of n variables, $X = \{X^1, X^2, \dots, X^n\}$; A is the set of all possible actions; T is a transition function, $T : S \times A \rightarrow \Delta(S)$, which specifies the probability distribution over the next states given the current state and action; C is a cost function, $C : S \times A \rightarrow \mathbb{R}$, which specifies the cost of performing each action from each state; H is the period of consideration over which the plan must be optimal, also known as the horizon, $0 < H \leq \infty$; and s_0 is the starting state of the process.

In order to gain insight into the functioning of MDPs, let us model the problem of composing the component WSCs of *Collect Insurance Information* composite service at level 1 of the mortgage scenario in Fig. 1 as a MDP. The state of the composition is captured by the random variables – **hazard and flood insurance information available, CTIC title insurance available, Delta insurance available, and TICORE title insurance available**. A state is then a conjunction of assignments of either *Yes*, *No*, or *Unknown* to each variable. Actions are WS invocations, $A = \{\text{Hazard and Flood Insurance Information, CTIC Title Insurance, Delta Insurance, and TICORE Title Insurance}\}$.

The transition function, T , models the non-deterministic effect of each action on some random variable(s). For example, invoking the WS *CTIC Title Insurance* will cause **CTIC title insurance available** to be assigned *Yes* with a probability of $T(\text{CTIC title insurance available} = \text{Yes} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$. This rate of order satisfaction depends on two probabilities: (1) the probability that CTIC is able to provide the insurance, and (2) the availability of the CTIC Title Insurance’s WS interface. If the two availabilities are independent of one another², we may view T as a product of these two probabilities: $T(\text{CTIC title insurance available} = \text{Yes} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = Pr(\text{CTIC title insurance product availability} = \text{Yes}) \times \text{WS Availability}$. Similarly, the **CTIC title insurance availability** will be assigned *No* with a probability of $T(\text{CTIC title insurance available} = \text{No} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = Pr(\text{CTIC title insurance product availability} = \text{No}) \times \text{WS Availability}$, and *Unknown* with a probability of $1 - \text{WS Availability}$.

²The two probabilities could be dependant on each other depending on the underlying business logic of the WS. For example, insurance availability may influence CTIC’s decision to keep the WS active. For simplicity in our scenario, however, we assume that they are independent.

ity. Note that the latter occurs when the WS fails or is not available.

The cost function, C , prescribes the cost of performing each action. Analogous to the calculation of the transition function T , C is some combination (e.g. a sum) of the cost of invoking the CTIC Title Insurance WS and the cost of the insurance itself. We let H be some finite value which implies that the broker is concerned with getting the most optimal WSC possible within a fixed number of steps. Since no information is available at the start state, all random variables will be assigned the value *Unknown*.

Once the mortgage broker has modeled its WS composition problem as a MDP, it may apply standard MDP solution techniques to arrive at an optimal process. These solution techniques revolve around the use of stochastic dynamic programming [18] for calculation of the optimal policy using *value iteration*:

$$V^n(s) = \min_{a \in A} Q^n(s, a) \quad (1)$$

where:

$$Q^n(s, a) = \begin{cases} C(s, a) + \sum_{s' \in S} T(s' | a, s) V^{n-1}(s) & n > 0 \\ 0 & n = 0 \end{cases} \quad (2)$$

where the function, $V^n : S \rightarrow \mathbb{R}$, quantifies the minimum long-term expected cost of reaching each state with n actions remaining to be performed, and $Q^n(s, a)$ is the action-value function, which represents the long-term expected cost from s on performing action a .

Once we know the expected cost associated with each state of the process, the optimal action for each state is the one which results in the minimum expected cost.

$$\pi^*(s) = \operatorname{argmin}_{a \in A} Q^n(s, a) \quad (3)$$

In Eq. 3, π^* is the optimal policy which is simply a mapping from states to actions, $\pi^* : S \rightarrow A$. The WSC is composed by performing the WS invocation prescribed by the policy given the state of the process and observing the results of the actions to obtain the next state (see Fig. 2).

Details of the algorithm for translating the policy to the WSC are given in [9].

Algorithm for generating WSC

Input: π^*, s_0

$s \leftarrow s_0$

while goal state not reached

$a \leftarrow \pi^*(s)$

Execute Web service a

Get response of a and construct next state, s'

$s \leftarrow s'$

end while

end algorithm

Figure 2: Algorithm for translating a policy into a WSC. Note the interleaving of WSC composition and execution.

3.2 Value of Changed Information

As discussed previously, the parameters of the participating services may change during the life-cycle of a WSC. For example, the cost of using the Delta Title Insurance provider’s

services may increase or the probability with which Delta can process a request may reduce. The former requires an update of the cost function, C , while the latter requires an update of the transition function, T , in the MDP model. In this article, we focus on a change in the transition function T , though our approach is generalizable to fluctuations in other model parameters as well.

Not all updates to the model parameters cause changes in the WSC. Furthermore, the change effected by the revised information may not be worth the cost of obtaining it. In light of these arguments, we need a method that will suggest a query, only when the queried information is *expected* to be sufficiently valuable to obtain. We provide one such methodology next.

As we mentioned before, we adopt a myopic approach to information revision, in which we query a single provider at a time for new information. For the **Insurance Information** composite service, this would translate to asking, say, only the **CTIC Title Insurance** provider for its current rates of request satisfaction (insurance and WS availability), as opposed to both the **CTIC Title Insurance** and **TICORE Title Insurance** providers, simultaneously. The revised information may change the following transition probabilities, $T(\text{CTIC title insurance available} = \text{Yes} \mid \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$, $T(\text{CTIC title insurance available} = \text{No} \mid \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$, and $T(\text{CTIC title insurance available} = \text{Unknown} \mid \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$.

Let $V_{\pi^*}(s|T')$ denote the expected cost of following the optimal policy, π^* , from the state s when the revised transition function, T' is used. Since the actual revised transition probability is not known unless we query the service provider, we average over all possible values of the revised transition probability, using our belief distribution over the values. These distributions may be provided by the service providers through pre-defined service-level agreements or they could be learned from previous interactions with the service providers. Formally,

$$EV(s) = \int_{\mathbf{p}} Pr(T'(\cdot|a, s') = \mathbf{p}) V_{\pi^*}(s|T') d\mathbf{p} \quad (4)$$

where $T'(\cdot|a, s')$ represents the distribution that may be queried and subsequently may get revised, $\mathbf{p} = \langle p_1, p_2, \dots, p_m \rangle$ represents a possible response to the query (revised distribution), m is the number of values that the variable under question may assume, and $Pr(\cdot)$ is our *belief* over the possible distributions. We illustrate using an example below:

As a simple illustration, let us suppose that we intend to query the **CTIC Title Insurance** WS provider for its current rate of order satisfaction. Eq. 4 becomes,

$$EV(s) = \int_{\langle p_1, p_2, 1-p \rangle} Pr(T'(\text{CTIC title insurance available} = \text{Yes/No/Unknown} \mid \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = \langle p_1, p_2, 1 - (p_1 + p_2) \rangle) V_{\pi^*}(s|T') d\mathbf{p}$$

given that the random variable **CTIC title insurance available** assumes either *Yes*, *No*, or *Unknown* on checking the status of the title insurer.

Let $V_{\pi}(s|T)$ be the expected cost of following the original policy, π , from state s in the context of the revised model parameter, T' . We recall that the policy, π , is optimal in the absence of revised information. We formulate the *value*

of *change* (VOC) due to the revised transition probabilities as:

$$VOC_{T'(\cdot|a, s')}(s) = \int_{\mathbf{p}} Pr(T'(\cdot|a, s') = \mathbf{p}) [V_{\pi}(s|T') - V_{\pi^*}(s|T')] d\mathbf{p} \quad (5)$$

The subscript to VOC , $T'(\cdot|a, s')$, denotes the revised information inducing the change. Intuitively, Eq. 5 represents how badly, on average, the original policy, π , performs in the changed environment as formalized by the MDP model with the revised T' .

We point out that the VOC shares its conceptual underpinnings with the value of perfect information (VPI) [19]. Indeed, both of them may be seen as special cases of the value of information idea, which determines whether new information is useful to a particular process. However, there is an important difference between the two concepts. VPI computes the value of *additional* information, while the VOC provides the value of *revised* information. We illustrate this distinction further in [14].

Analogous to VPI, the following theorem holds for VOC, whose proof is in [14].

THEOREM 1. $\forall s \in S, \quad VOC(s) \geq 0$ where $VOC(\cdot)$ is as defined in Eq. 5.

Querying for information from service providers may often be tedious, time consuming and subsequently, expensive. The expenses could include, for example, contractual costs and intangible costs such as the delay incurred while awaiting the revised information. We must therefore undertake the querying only if we expect it to pay off. In other words, we query for revised information from a state of the WSC only if the VOC due to the revised information in that state is greater than the query cost. More formally, we query if

$$VOC_{T'(\cdot|a, s')}(s) > QueryCost(T'(\cdot|a, s'))$$

where $T'(\cdot|a, s')$ represents the distribution we want to query.

4. HIERARCHICAL WEB SERVICE COMPOSITION WITH VOC

In order to promote scalability, WSCs may often be decomposed into a hierarchy. In particular, a WSC may include component services that are themselves WSCs. Such a nesting could be repeated down to any level, *ad infinitum*. We refer to a WS that is itself implemented as a lower level WSC as a composite WS. Before we present an approach that queries WSs guided by VOC in a hierarchical WSC, we need a way to compose the WSC because we interleave adaptation with composition and execution.

In [23], Zhao and Doshi provide a method of composition that exploits a hierarchical decomposition. We slightly modify this approach and model each level of the hierarchy using a MDP. Specifically, the lowest levels of the hierarchy (leaves) are modeled using a MDP containing *primitive* actions, which are invocations of the WSs. Higher levels of the composition problem are modeled using MDPs that contain *abstract* actions, which represent the execution of lower level WSCs. While formulating the lowest level MDPs is straightforward and proceeds as in Section 3.1, we must derive the parameters of the composite WS to permit the formulation of the MDP that models the composition problem at the higher level WSCs.

4.1 Parameters for Composite Web Services

Model parameters for abstract action A higher level MDP is so far not well defined because meaningful parameters for the abstract actions in the model are not given. For example, in the mortgage scenario of Fig. 1, the composite WS, **Insurance Information**, at level 1 is composed of primitive WSs: **Hazard and Flood Insurance Information** and one or multiple WSs among **CTIC Title Insurance**, **Delta Title Insurance** and **TICORE Title Insurance**. Transition probabilities associated with the abstract action **Collect Insurance Information** are not available, but instead must be derived from the transition probabilities associated with the primitive actions.

Zhao and Doshi [23] utilize the correspondence between the high level abstract actions and the corresponding low level primitive actions. Let the abstract action, \bar{a} , represent the sequential execution, in some order, of primitive actions, $\{a_1, a_2\}$, of the underlying primitive MDP. Because the order in which the primitive actions a_1 and a_2 are performed is not known from beforehand, there may be multiple ways to achieve the composition – start from the state s_p and reach the state, s_e . Let $s_p \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_e$ be one such path, where s_1 is an intermediate state of the WSC, then, $T(s_1|a_1, s_p) \times T(s_e|a_2, s_1)$ is the probability of following this path, where T is the transition function of the primitive MDP. The required probability, $Pr(s_e|s_p)$, is the sum of the probabilities of following all such paths. Analogously, the cost of performing the abstract action is the average of the cost of following each of the possible paths that achieve the composition weighted by the probability of that path.

Algorithm for approximate PDF over product space

```

Input:  $\mathcal{N}(\mu_1, \sigma_1), \mathcal{N}(\mu_2, \sigma_2)$ 

n //number of samples
numBins //number of bins used in histogram, cdf and pdf
frequencyCountBins[1..numBins] //freq. count for histogram
cdf[1..numBins], pdf[1..numBins]
Sample densities and tabulate freq. of product of samples
for i = 1 to n
  Sample  $s_1 \sim \mathcal{N}(\mu_1, \sigma_1)$ 
  Sample  $s_2 \sim \mathcal{N}(\mu_2, \sigma_2)$ 
   $p \leftarrow s_1 \times s_2$ 
  j  $\leftarrow$  bin corresponding to p
  increment frequencyCountBins[j] by 1
end for
Convert to a cdf
for i from 2 to numBins
   $cdf[i] \leftarrow cdf[i - 1] + frequencyCountBins[i]$ 
end for
Convert from cdf to a pdf
for i from 2 to numBins
  Use inverse of trapezoidal rule for numerical quadrature
  with values in  $cdf[i]$  and  $cdf[i - 1]$  to find  $pdf[i]$ 
End for
end algorithm

```

Figure 4: Sampling algorithm for approximating a probability density over a product of two independent random variables with Gaussian distributions.

Belief over volatile parameters of composite WS We may model the mortgage broker’s beliefs over the possible parameters of the WS, $(Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 5) using density functions. We let the densities for the WSs take the form of *Gaussian* density functions³. Fig 3 show examples

³Note that other density functions (such as betas and polynomials) may also be used

of such densities, defined for the Hazard and Flood Insurance Information WS (Fig. 3(a)) and the Title Insurance WSs (Fig. 3(b)). Other WSs in the mortgage loan process are assumed to have analogous densities. We emphasize that these densities are marginalizations of the more complex ones that would account for all the factors that may influence, for example, a service’s rate of request satisfaction.

Means of the densities reveal that the Delta Title Insurance WS tends to be less reliable in satisfying requests than the other title insurers’ WSs. Note also that the Hazard and Flood Insurance Information WS is very reliable, as its mean is close to 1 and its standard deviation is relatively small.

Modeling beliefs over the volatile parameters of the composite services is more complex. Previously, we mentioned that the transition function of the composite service may be obtained by taking the product of the transition probabilities of the corresponding individual services. We use this in forming beliefs over the volatile parameters of composite services. For example, let us obtain the belief for the **Insurance Information** composite WS at level 1. The availability of the **Insurance Information** service is the sum of the products of the availability of **Hazard and Flood Insurance** service and one or more of the different **Title Insurance** services. *Therefore, the belief density over the availability of Insurance Information will be the summation of functions over the product space: availability of Hazard and Flood Insurance WS \times availability of a Title Insurance service.*

Although we model the densities over availability of individual WSs as Gaussians (for example Figs. 3(a) and (b)), the function over the product space is not a Gaussian but rather a modified Bessel function of the second kind [10]. Because generating the Bessel function exactly is complex, we utilize a sampling method to generate the density over the given product space, which converges to the exact as the number of samples approaches infinity. The algorithm for the sampling approximation is given in Fig. 4. We first sample the individual densities and tabulate the frequencies of the product of the two independent variables. The frequency histogram is converted into a normalized cumulative distribution function (cdf), which may be converted into an approximate probability density function. We show an approximate density obtained by the algorithm in Fig. 3(c). Densities analogous to this one are summed to obtain the broker’s belief over the volatility of aggregate parameter of the composite WS **Insurance Information**.

4.2 Algorithm

We make a small change to the algorithm outlined in Fig. 2 to utilize VOC. In order to formulate and execute the WSC, we simply look up the current state of the WSC in the policy and execute the WS prescribed by the policy for that state. The response to the WS invocation determines the next state of the WSC. We adapt the composition to consider fluctuations in the model parameters by interleaving the formulation with VOC based selective querying.

For each state encountered during the execution of the WSC at some level l , we find the WS whose revised information is expected to bring about the greatest change in the WSC. In other words, we select the provider associated with the WS invocation, a , to possibly query for whom the VOC is maximum:

$$a^* = \underset{a \in A}{\operatorname{argmax}} \operatorname{VOC}_{T'(\cdot|a, s')}(s) \quad (6)$$

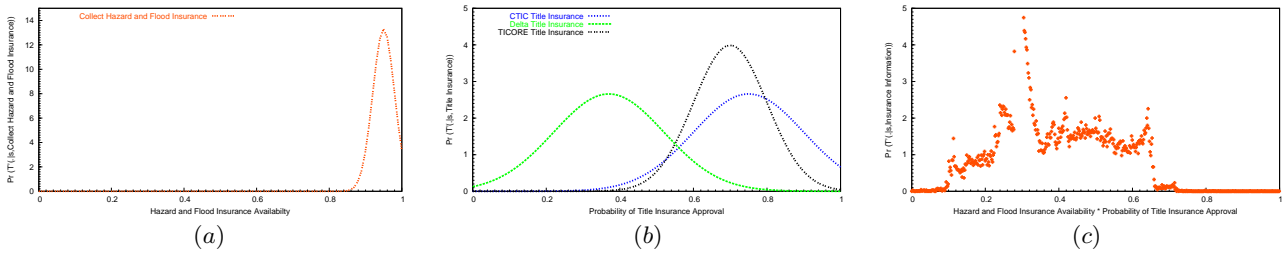


Figure 3: (a) Probability density function representing the mortgage broker’s beliefs over the Hazard and Flood Insurance WSs’ probabilities of satisfying requests. (b) Probability density functions representing the mortgage broker’s beliefs over the Title Insurance WSs’ probabilities of satisfying requests. (c) The resulting approximate probability density function for the composite WS Insurance Information.

Algorithm for adaptive Web service composition – AWSC

Input:
 $\langle \pi_l^*, \pi_{l-1}^*, \dots, \pi_0^* \rangle$ //optimal policies
 s_0 //initial state
 l //depth
 H //horizon

$s \leftarrow s_0$
 $n \leftarrow H$

1. **while** $n > 0$
2. **if** $VOC^*(s) > QueryCost(T'(\cdot|a^*, s'))$
3. **if** a^* is composite
4. $a_k^* \leftarrow findWS(l-1, A, s)$
5. Query a_k^* //primitive WS
6. $T' \leftarrow UpdateModel(Level\ l, Level\ k)$
7. Calculate policy π_l^* using the new MDP with T'
8. $a \leftarrow \pi_l^*(s)$
9. **if** a is composite
10. Recursively call AWSC for a at level $l-1$ and π_{l-1}^*
11. **else**
12. Execute primitive Web service a
13. Get response of a and construct next state s'
14. $s \leftarrow s'$
15. $n \leftarrow n-1$
16. **end while**

Figure 5: Algorithm for executing and adapting a hierarchical WSC to revised information.

Let $VOC^*(s)$ represent the corresponding maximum VOC. We may then obtain $VOC^*(s)$ as follows:

$$VOC^*(s) = \max_{a \in A} VOC_{T(\cdot|a, s')}(s) \quad (7)$$

The algorithm for an adaptive WSC is shown in Fig. 5. If a^* is a composite WS at level l , we must find the WS at level $l-1$ to query (lines 3-5). This procedure recurses down the nesting level until we select a primitive WS to query. We outline this recursive procedure in Fig. 6.

After querying a_k^* , where k represents the level at which the queried WS resides, we must formulate a new transition, T' , and policy, π^* , for the level k WSC. Subsequently, a new transition function (for the corresponding composite WS) and policy must be computed at all levels up to the top most level, l (lines 6-7). For example, if the CTIC Title Insurance is queried, we reformulate the policy at level 0 given the revised information and recompute the aggregate parameters of the composite WS, Insurance Information, at level 1. We subsequently revise the transition function, T' and resolve π^* at level 1. This recursive procedure is presented in Fig. 7.

Algorithm for findWS $^*(k, A, s^k)$

Input :
 k //level
 A //action set
 s^k //state at level k

1. $a_k^* \leftarrow \underset{a \in A}{argmax} VOC_{T(\cdot|a, s')}(s_k)$
2. **if** a_k^* is a primitive service **then**
3. return a_k^*
4. **else** // a_k^* is a composite service
5. $s^{k-1} \leftarrow$ initial state of the WSC at level $k-1$
6. return $findWS(k-1, A, s^{k-1})$

end algorithm

Figure 6: Function $findWS$ recursively finds a WS that yields the highest VOC.

Algorithm for UpdateModel(l, k)

Input :
 k //depth
 l //current level

1. **if** $l > k$ **then**
2. $T' \leftarrow UpdateModel(l-1, k)$
3. Calculate new policy π_{l-1}^* using the MDP with T'
4. Formulate new T_c for composite WS given T'
5. return T_c
6. **else**
7. Integrate revised information from query to form T'
8. Calculate new policy π_k^* using the MDP with T'
9. return T'

end algorithm

Figure 7: Recursively update the transition probabilities and policies in the hierarchical composition using the revised information.

4.3 Computational Complexity

We derive the complexity of the algorithm in Fig. 5 next.

THEOREM 2. Let N denote the number of possible values of a random variable X_i , H denote the horizon, $|A|$ denote the largest number of services available at any of the composition levels, and l denote the depth of the hierarchy. The worst-case complexity of the algorithm in Fig. 5, is:

$$\mathcal{O}((l+2) \cdot (N^{2|X|} |A|^2 H^{l+1}) + l \cdot |A|^H H^l)$$

PROOF. The outer loop (line 1 of Fig. 5) will terminate when the composition has completed (taking at most H steps). Within the body of the loop we focus on four operations in particular. First, we find $VOC^*(s)$ as shown in line

2. Previous work [15] showed that the worst-case complexity of this operation is $\mathcal{O}(N^{2|X|}|A|^2H)$. Second, in line 4, the recursive function $findWS$ (see Fig. 6) is called when the WS a^* corresponding to $VOC^*(s)$ is a composite service. $findWS$ will be recursively called until a primitive WS is found for querying. Time complexity of $findWS$ is obtained from the following recurrence relation:

$$T(l) = T(l-1) + \mathcal{O}(N^{2|X|}|A|^2H) \quad (8)$$

where $T(0) = \mathcal{O}(N^{2|X|}|A|^2H)$. The first term, $T(l-1)$, is due to the recursion in levels of the hierarchy and the second term, $\mathcal{O}(N^{2|X|}|A|^2H)$, is due to the VOC computation in the algorithm. In the worst case, the procedure will traverse through the entire depth of the composition. At each level, the algorithm computes $VOC^*(s)$ (line 1) taking $\mathcal{O}(N^{2|X|}|A|^2H)$ time. Solving the recurrence will take $\mathcal{O}((l+1) \cdot (N^{2|X|}|A|^2H))$ time. Third, in line 6, the main algorithm calls the recursive function $UpdateModel$ (Fig. 7). We may write the recurrence relation defining its complexity as follows:

$$T(l) = T(l-1) + \mathcal{O}(|A|^H + N^{2|X|}|A|H) \quad (9)$$

where the recursion traverses down to level k and $T(k) = \mathcal{O}(N^{2|X|}|A|H)$. At each level of the composition till k , we form the new policy π_{l-1}^* , which incurs $\mathcal{O}(N^{2|X|}|A|H)$. We then reformulate T_c , which requires $\mathcal{O}(|A|^H)$ time to compute all possible paths. The cost for these operations is shown in the second term of Eq. 9. We solve this recurrence to obtain $\mathcal{O}((l+1) \cdot N^{2|X|}|A|H + l \cdot |A|^H)$. Finally, line 10 of Fig. 5 is a recursive call to the $AWSC$ algorithm itself. In the worst case, the loop will be executed H times at each level of the composition. We may write this recurrence as:

$$T(l) = H \cdot T(l-1) + \mathcal{O}((l+1) \cdot (N^{2|X|}|A|^2H^2) + l \cdot |A|^H H) \quad (10)$$

where $T(0) = \mathcal{O}(N^{2|X|}|A|^2H^2)$ - complexity of obtaining the policy. Each iteration of the loop goes through all three of the previously described operations. The time needed is the sum of the times needed for these operations: $\mathcal{O}(N^{2|X|}|A|^2H^2) + \mathcal{O}((l+1) \cdot (N^{2|X|}|A|^2H)) + \mathcal{O}((l+1) \cdot N^{2|X|}|A|H + l \cdot |A|^H)$, which becomes $\mathcal{O}((l+2) \cdot (N^{2|X|}|A|^2H) + l \cdot |A|^H)$. This forms the second term of Eq. 10.

We solve Eq. 10 to obtain the complexity of the algorithm in Fig. 5: $\mathcal{O}((l+2) \cdot (N^{2|X|}|A|^2H^{l+1}) + l \cdot |A|^H H^l)$ \square

Notice that the complexity is exponential in l . We emphasize that this is the worst-case complexity, and in practice, we do not expect each step of a WSC to be nested.

5. EXPERIMENTAL RESULTS

We first outline our SOA, in which we wrap the VOC computations in WSDL based internal WSs, followed by our experimental results on the performance of the adaptive WSC. The results were compared to approaches that use a static, unchanging policy, query random WSs and other heuristics.

5.1 Architecture

The algorithm described in Fig. 5 is implemented as a WS-BPEL⁴ flow and all WSs were implemented using WSDL⁵.

⁴WS-BPEL v2: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

⁵WSDL v2: <http://www.w3.org/TR/wsd120/>

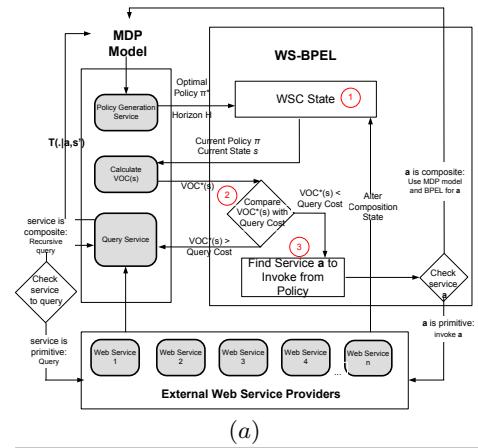


Figure 8: SOA for implementing our adaptive WSC. (a) demonstrates the interaction of the composition with internal services. In (b), we show a portion of the WS-BPEL for the mortgage acquisition process. Labels (1) (2) and (3) in (a) correspond to its associated markup in (b).

To the WS-BPEL flow, we give the optimal policies, $\langle \pi_l^*, \pi_{l-1}^*, \dots, \pi_0^* \rangle$, of the top level WSC, the start state, and horizon as input. Our experiments utilized IBM's BPWS4J engine for executing the BPEL process and AXIS 2.0 as the container for the WSs. We show our SOA in Fig. 8(a).

Within our SOA, we provide internal WSs that solve the MDP model of the composition problem and generate the policy, and for computing the VOC. If the VOC exceeds the cost of querying a particular service provider (this cost is also provided as an input), the WS-BPEL flow invokes a special WS whose function is to query the service provider's information-providing WSs for revised information. This

information is used to formulate and solve a new MDP and the output policy is fed back to the WS-BPEL flow. This policy is used by the WS-BPEL flow to invoke the prescribed external WS if it is not a composite one, and the response is used to formulate the next state of the process. If the WS to invoke is composite, the procedure is recursively repeated for the lower level WSC. This procedure continues until the goal state is reached or the total number of steps are exhausted.

As we utilize WS-BPEL in a somewhat non-standard way, we provide some details on how we implement the WS-BPEL flow in Fig. 8(b). First note that the following constructs are added to implement the VOC based algorithm:

- state and policy data structures,
- tasks that will invoke a VOC computation service, compare the VOC to a given query cost, and regenerate the policy if needed, and
- a task that will invoke the external services providers as recommended by the policy.

As outlined by section (1) in Fig. 8(a), state is stored in the BPEL document by creating a complex message type, *stateMessage* and stored in the *stateData* variable. Similarly, complex message type *policyMessage*, is stored in the *policy* variable, and used to represent the given policy.

The `< while >` condition corresponds to the while loop in Fig. 5. Each state has an associated `< switch >` `< case >` construct. In each `< case >`, the WSC invokes the **Calculate VOC** WS, which upon completion returns VOC^* (section (2)). The VOC^* value is compared to a *QueryCost* variable. If the returned VOC^* is greater than the *QueryCost*, then the associated service is queried. The revised parameters are integrated into the MDP, which invokes the policy generator service, returning the new optimal policy(s) thereby replacing the old policy of the WSC. The policy is then used to recommend the optimal service to invoke in the state. This process repeats until the composition has terminated (i.e. all steps have been exhausted).

5.2 Performance Evaluation

We utilized the mortgage loan processing scenario (Section 2) for our evaluations. We simulated querying the different WSs for their current percentage of request satisfaction (availability).⁶

We model the mortgage broker’s beliefs over the volatile parameter of the individual WSs, ($Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 5) using *Gaussian* density functions. For composite WSs, we derive the densities over the aggregated parameters as shown in Section 4.1.

In Fig. 9(a), we compare the VOC-driven selective querying with four other strategies with respect to the average cost incurred from the execution of the adapted hierarchical WSCs, as the cost of querying the WSs for information is increased. Our methodology consisted of running a trial of 150 independent instances of each composition within a simulated volatile environment, where the queried parameters of the services were distributed according to the corresponding density plots (see Fig. 3). We ensured that the compositions using each of the five strategies received similar responses from the services.

The four other approaches utilized were:

1. **Static policy** This is our baseline approach that ignores adaptation and the initial policy is utilized unchanged for executing the composition in each instance.
2. **Random query** In this approach, we randomly select a service at each time step to query for revised information.
3. **Intermittent querying** We begin by querying services every alternate instance and as the costs of querying increase, we reduce the frequency with which we query services.
4. **Largest difference** This approach utilizes the distributions of the services parameters shown in Fig. 3. It selects a service to query whose existing parameter value is most different from the mean as obtained from the corresponding distribution.

We note that each of the approaches mentioned above are naive analogies of certain aspects of the VOC based approach. Thus, the approaches provide an effective testbed with which to compare our VOC based WSC adaptation.

Intuitively, as we increase the cost of querying, the VOC based approach performs less queries and adapts the WSC less. For large query costs, its performance is similar to using a WSC with an unchanging policy. For smaller query costs, a VOC based approach will query frequently, though not as much as a strategy that always queries some provider, such as *random query*. As we increase the query costs, the VOC based approach will allow a query for revised information only if its value exceeds the cost. Though *intermittent querying* naively seeks to emulate this behavior, it performs worse because it does not utilize the value of a potential change in the composition in deciding when to query. We note that the *largest difference* approach performs well for lower query costs, though worse than the VOC based approach. This is because the service exhibiting the largest difference from the mean in its parameter value is often the one that brings about the largest change in the composition. However, this is not always the case – for example, a large change in the parameter of a mandatory service, such as the **Credit Check** service in the mortgage process, does not affect the composition though the approach will query it and incur the query cost. In addition, the difference in parameter values may not be comparable to query costs. In summary, a WSC that is adapted using VOC performs better (incurs less average cost) because only significant changes to the WSC are carried out while simultaneously avoiding frequent costly queries.

While it is impossible to guarantee *a priori* that all issued queries will result in changes in the WSC, we measure the percentage of queries that do not change the WSC – we refer to such queries as false positives. As we show in Fig. 9(b), VOC based querying results in significant less false positives compared to the other strategies. The false positives drop to zero as the query cost increases because a query is issued only if the VOC exceeds the cost of querying. Other comparative approaches are independent of the query costs. Notice that randomly querying results in approximately 75% of the queries being false positives. We observe that the reduced percentage of false positive queries is responsible, in part, for the reduced cost of adapting WSCs using VOC.

We point out that adaptation using VOC comes at a computational price. In Table 1 we give the run times for each

⁶Of course, the rate of request satisfaction, for example, would depend on the amount of the loan and other factors; we assume that these will be provided.

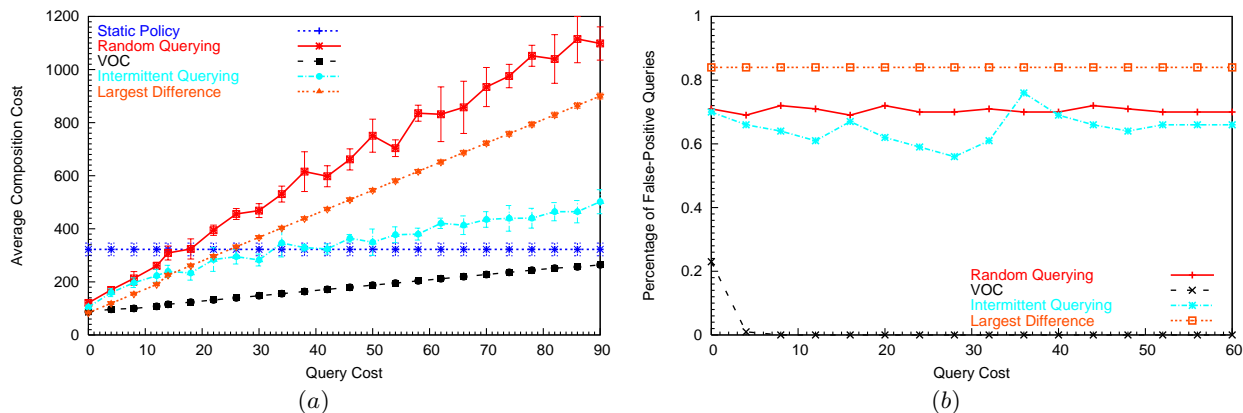


Figure 9: Comparisons of the VOC based adaptive WSC with the static policy and other querying approaches for the mortgage loan acquisition. (a) We measure the average cost. Lower cost indicates better performance. (b) We measure the percentage of false-positive queries. Notice that VOC results in a WSC that is most cost efficient among all strategies, in part, because it issues a much lower percentage of queries that turn out to be false positives.

Query strategies	Time (ms)		
	Query cost = 0	Query cost = 40	Query cost = 80
Static Policy	609 ± 50	609 ± 50	613 ± 47
Random query	759 ± 72	755 ± 66	759 ± 72
VOC	2112 ± 196	1800 ± 136	1650 ± 116
Intermittent querying	653 ± 4	540 ± 20	470 ± 20
Largest difference	535 ± 8	535 ± 12	541 ± 11

Table 1: Running times of the various querying strategies for different query costs. We used Linux platform on Xeon 3.8GHz with 2.0GB memory.

of the comparative approaches used previously. We first observe that an adaptive WSC that uses VOC runs two to three times slower than a WSC that does not adapt (static policy). However, this time difference reduces when other strategies for adapting WSCs are utilized. The additional runtime of the VOC is primarily due to the computations required for Eq. 5. To be realistic, we included a short lag time in receiving the query response from the WSs. Notice, however, that as the query cost increases, the time required by the VOC based approach decreases, because it issues less queries.

6. RELATED WORK

Dynamism manifests in WSC environments in a variety of ways. Indeed, research into adaptation has been broad and encompasses many different types of volatility. Our work focuses on data volatility, which van der Aalst describes as dynamism in the information perspective [20]. Only recently have researchers turned their efforts toward dealing with this type of dynamism.

Chafle et al. [5] pre-specify several alternate plans at the logical level, physical level, and the runtime level. Depending on the type of changes in the environment, alternative plans from these three stages are selected. While capable of adapting to several different events, many of the alternative pre-specified plans may not be used making the approach inefficient. Further, there is no guarantee of optimality of the resulting WSCs. Paques et al. [17] address changes by

creating a WS “adaptation space”. The adaptation space represents alternative logical WS compositions that may be used if a previous composition instance fails or is found to be suboptimal. While the adaptation space allows WSCs to adapt to changes in the data, it does not consider the costs of obtaining the revised data. Doshi et al. [9] adapt compositions using a technique that manages the dynamism of WSC environments through Bayesian learning. WSC model parameters are updated based on previous interactions with the individual WSs and the composition plan is regenerated using these updates. This method suffers from being slow in updating the parameters, and the approach may result in plan (process flow) re-computations that do not bring about any change in the WSC. Au et al. [2] introduce a framework that composes processes in the presence of data volatility. Using a reactive querying policy, they obtain current parameters of the WSC by querying WS providers only when the parameters expire. While this is similar in concept to our approach, plan re-computation is assumed to take place irrespective of whether the revised parameter values are expected to bring about a change in the composition. This may lead to frequent unnecessary computations. Gotz and Mayer-Patel [12] incorporate multidimensional data adaptation using a metric similar to the value of information to determine if new information may impact the utility of their application. The authors, however, primarily focus on multimedia applications rather than business processes.

Nanjangud et al. [16] and Charfi et al. [7] apply aspect-

oriented programming to WSCs. Aspects were used to adapt to changes in WS components dynamically and consistently. Analogous to the traditional exception-handling techniques, this line of work focuses on composition correctness and consistency. Gomadam et al. [11] utilize semantic associations to identify events that may cause changes in a WSC. The focus, however, is on event identification as a precursor to adaptation. In a somewhat different vein, Verma et al. [21] and Wu and Doshi [22] explore adaptation in WSCs in the presence of coordination constraints between different WSs. This line of work is complementary as we do not consider such constraints here.

7. CONCLUSION

Real world process environments are volatile – non-functional parameters of the services such as costs and reliability may vary over time. In such environments, WSCs must adapt to the revised information to remain cost effective. Previously, the value of changed information has been proposed to gauge the value of the expected change that revised information may bring to the WSC, and it is compared with the cost of obtaining the information. If the probable revised information is worth the cost of obtaining it, the providers are queried for their WS’s current parameters and we reformulate the WSC using the revised information. While previously, VOC was applied toward adapting simple flat WSCs, in this paper, we extended its applicability to hierarchical WSCs. This is significant because large WSCs often tend to have a hierarchy. Two of the primary challenges that we address are how to obtain beliefs over volatility of composite WS parameters, and which of the component WSs to invoke if a composite WS is found to potentially cause the most expected change in the composition.

The focus of our future work will be to improve upon our current model of volatility of a WSC environment. This will enable the development of more efficient approaches for adaptation. We also intend to focus on non-myopic approaches, which may enhance querying strategies that use VOC. Furthermore, we will investigate approximate ways of introducing revised values to the model, when applicable, so as to avoid full recomputations of the policies.

8. REFERENCES

- [1] V. Agarwal, G. Chafle, K. Dasgupta, N. Kamik, A. Kumar, S. Mittal, and B. Srivastava. Synth: A system for end to end composition of web services. *Journal of Web Semantics*, 3:311–339, 2005.
- [2] T.-C. Au and D. Nau. Reactive query policies: A formalism for planning with volatile external information. In *CIDM*, pages 243–250, 2007.
- [3] A. Borgida and T. Murata. Tolerating exceptions in workflows: a unified framework for data and processes. In *WACC*, pages 59–68, 1999.
- [4] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1:281–308, 2004.
- [5] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava. Adaptation in web service composition and execution. In *ICWS*, pages 549–557, 2006.
- [6] G. Chafle, P. Doshi, J. Harney, S. Mittal, and B. Srivastava. Improved adaptation of web service compositions using value of changed information. In *ICWS*, pages 784–791, 2007.
- [7] A. Charfi and M. Mezini. Aspect-oriented web service composition with ao4bpel. In *ECOWS*, pages 168–182, 2004.
- [8] N. Desai, A. Chopra, and M. Singh. Business process adaptations via protocols. In *SCC*, pages 601–608, 2006.
- [9] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using markov decision processes. *Journal of Web Services Research*, 2(1):1–17, 2005.
- [10] A. Glen, L. Leemis, and J. Drew. Computing the distribution of the product of two continuous random variables. *Computational Statistics & Data Analysis*, 44(3):451–464, 2004.
- [11] K. Gomadam, A. Ranabahu, L. Ramaswamy, A. Sheth, and K. Verma. A semantic framework for identifying events in a service oriented architecture. In *ICWS*, pp. 545–552, 2007.
- [12] D. Gotz and K. Mayer-Patel. A general framework for multidimensional adaption. In *ICME*, pages 612–619–126, 2004.
- [13] A. S. Y. Han and C. Bussler. A Taxonomy of Adaptive Workflow Management. In *CSCW Workshop on Towards Adaptive Workflow Systems*, 1998.
- [14] J. Harney and P. Doshi. Adaptive web processes using value of changed information. In *ICSOC*, pages 179–190, 2006.
- [15] J. Harney and P. Doshi. Speeding up adaptation of web service compositions using expiration times. In *WWW*, pages 1023–1032, 2007.
- [16] N. Narendra, K. Ponnalagu, J. Krishnamurthy, and R. Ramkumar. Run-time adaptation of non-functional properties of composite web services using aspect-oriented programming. In *ICSOC*, pages 546–557, 2007.
- [17] H. Paques, L. Liu, and C. Pu. Adaptation space: A design framework for adaptive web services. *International Journal of Web Service Research*, 1(3):1–24, 2004.
- [18] M. Puterman. *Markov Decision Processes*. John Wiley & Sons, NY, 1994.
- [19] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.
- [20] W. M. P. van der Aalst and S. Jablonski. Dealing with workflow change: identification of issues and solutions. *International Journal of Computer Systems Science and Engineering*, 15(5):267–276, September 2000.
- [21] K. Verma, P. Doshi, K. Gomadam, J. Miller, and A. Sheth. Optimal adaptation in web processes with coordination constraints. In *ICWS*, pages 257–264, 2006.
- [22] Y. Wu and P. Doshi. Regret-based decentralized adaptation of web processes with coordination constraints. In *SCC*, pages 262–269, 2007.
- [23] H. Zhao and P. Doshi. Haley: A hierarchical framework for logical composition of web services. In *ICWS*, pages 312–319, 2007.