

Selective Querying for Adapting Hierarchical Web Service Compositions Using Aggregate Volatility

John Harney *and* Prashant Doshi
LSDIS Lab, University of Georgia
Athens, GA, 30602
{jfh, 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. In this paper, we present an approach for utilizing VOC in the context of a WSC composed of services and lower level WSCs, which induces a natural hierarchy over the composition.

1. Introduction

Approaches for composing Web services (WS) 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. Solutions have been presented to address some of these changes, ranging from exception handling techniques defined in [2] to instituting protocol adaptations in [5].

In order to adapt, WS compositions (WSC) must possess up-to-date knowledge of the revised information that could be obtained by querying the component services (or providers) for the revised parameters. Querying for service parameters, however, comes with its own attendant challenges. Changes to the parameters do not necessarily warrant a change to the WSC. Additionally, WSCs may operate over an open and large scale system making querying tedious and costly. Queries must therefore be carefully managed – specifically, the adaptive WSC should know: (1) when it is cost effective to query for the changed information and, (2) which service(s) to query.

Harney and Doshi [9] proposed a method for selective

querying using the *value of changed information (VOC)*. In particular, the approach computes the trade-off between the cost of querying for up-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.

While previously the usefulness of VOC was demonstrated in the context of simple WSCs, in this paper, we focus on a *hierarchical* WSC. In many cases, 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 QoS parameters of component WSs exist [3, 14], 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 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 our approach performs well in the presence of data volatility, reducing the costs in comparison to other query approaches.

2 Scenario: Mortgage Loan Processing

Our scenario is a simplified version of a mortgage loan acquisition process, typically used by brokers that service mortgage loans to individual clients. Our 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.

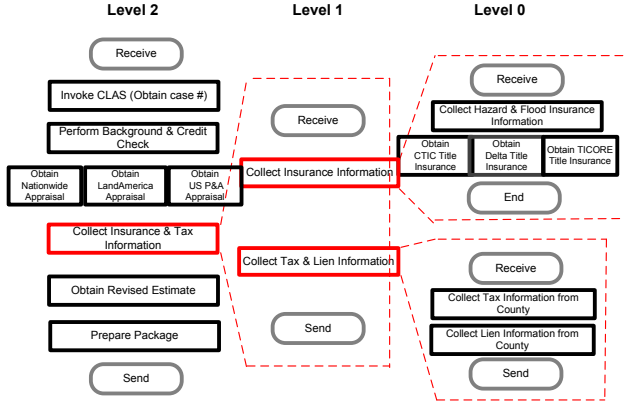


Figure 1. A hierarchical mortgage loan process used by a mortgage broker.

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 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 the client, utilize the WS of the credit rating agencies to perform financial background and credit checks, and hire an appraisal agency for home appraisal. In this example, the broker may choose between three different appraisal services: Nationwide, LandAmerica, and US P and A. A choice 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 home insurance, tax and lien information.

We formulate the Insurance and Tax Information service (shown in red) as a composite service. The corresponding level 1 composition consists of two services for collecting insurance information and collecting tax and lien information. These services are themselves composite. For instance, the Insurance Information service gathers information on both hazard and title insurances. The broker must decide on a vendor (CTIC, Delta, or TICORE) to provide the title insurance. 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.

3 Background

For the purpose of illustration, we select a decision-theoretic planning technique for composing WSs [6]. We

briefly review the definition and formulation of the *value of changed information* (VOC) for participating WSs, and refer the reader to Harney and Doshi [9] for more details.

3.1 Web Service Composition Using MDP

MDPs model the composition environment using a sextuplet $WP = (S, A, T, C, H, s_0)$, where $S = \prod_{i=1}^n X^i$ 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 possible actions; T is a transition function, $T : S \times A \rightarrow \Delta(S)$, specifying 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.

As an example, let us model the problem of composing the component WSs of the *Collect Insurance Information* composite service at level 1 of the mortgage scenario as a MDP. The state of the WSC is captured by the random variables – **hazard and flood insurance information available**, **CTIC title insurance available**, **Delta insurance available**, and **TICORE title insurance available** – and is 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 cost function, C , models the cost of invoking a service. 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 $T(\text{CTIC title insurance available} = \text{Yes} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$.

Once the mortgage broker has modeled its WSC problem as a MDP, it may apply standard solution techniques such as *value iteration* [11] for calculation of the optimal policy, $\pi^* : S \rightarrow A$. 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.

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 services may increase (requiring an update of C) or the probability with which Delta can process a request may reduce (requiring an update of T). In this paper, we focus on a change in T , though our approach is generalizable to fluc-

tuations in other model parameters as well. VOC [9] employs a *myopic* approach to information revision, in which we query a single provider at a time for new information.

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. 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. Since T' 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. We formulate the VOC as:

$$VOC_{T'(\cdot|a,s')}(s) = \int_{\mathbf{p}} Pr(T'(\cdot|a,s') = \mathbf{p}) [V_{\pi}(s|T') - V_{\pi^*}(s|T')] d\mathbf{p} \quad (1)$$

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. The subscript to VOC , $T'(\cdot|a,s')$, denotes the revised information inducing the change. Intuitively, Eq. 1 represents how badly, on average, the original policy, π , performs in the changed environment as formalized by the MDP model with the revised T' .

Querying for information from service providers may be expensive. Therefore, we query for revised information only if the VOC due to the revised information in a state is greater than the query cost. More formally, we query if $VOC_{T'(\cdot|a,s')}(s) > QueryCost(T'(\cdot|a,s'))$, where $T'(\cdot|a,s')$ is the distribution we want to query.

4 Hierarchical WS Composition with VOC

For manageability, WSCs are often decomposed into a hierarchy. In particular, a WSC may include component services that are themselves WSCs. 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.

Multiple approaches for composition exist that exploit a hierarchical decomposition [13, 14]. We slightly modify the approach of Zhao and Doshi [14] 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 higher level MDP.

4.1 Parameters for Composite WSs

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 [14] 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 candidate 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.

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. 1) using density functions. We let the densities for the WSs take the form of *Gaussian* density functions¹. Fig. 2 shows examples of such densities. Other WSs in the mortgage loan process are assumed to have analogous densities.²

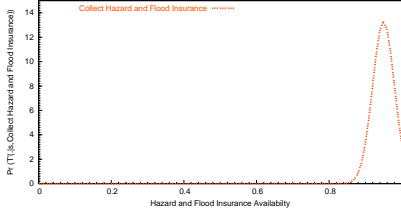
Means of the densities reveal that the Delta Title Insurance WS tends to be less reliable in satisfying requests than the other title insurers’ WSs. Note also that the Hazard and Flood Insurance Information WS is very reliable, having a mean close to 1 and standard deviation relatively small.

Modeling beliefs over the volatile parameters of the composite services is more complex. Previously, we mentioned that the transition function of the composite service

¹Note that other density functions (such as betas and polynomials) may also be used.

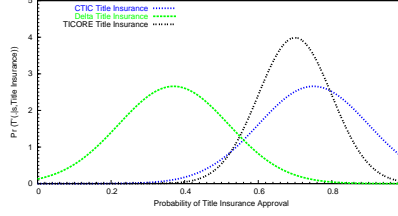
²We emphasize that these densities are marginalizations of the more complex ones that would account for all the factors that may influence, for example, a service’s rate of request satisfaction.

Hazard and Flood Insurance WS



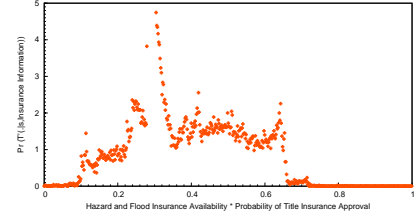
(a)

Title Insurance WS



(b)

Insurance Information Composite WS



(c)

Figure 2. (a) Probability density function (pdf) representing the mortgage broker’s beliefs over the Hazard and Flood Insurance WSs’ probabilities of satisfying requests. (b) Pdfs representing the broker’s beliefs over the Title Insurance WSs’ probabilities of satisfying requests. (c) The resulting approximate pdf for the composite WS Insurance Information.

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.*

Deriving beliefs for flow patterns Composite services such as the Insurance Information WS contain services executed in a strictly *sequential* path. Many WSCs, however, exhibit other flow patterns. Therefore, we must refine our derivation of the beliefs of composite services to accommodate these constructs. Let us consider a composition of two atomic services, w_1 and w_2 , having Gaussian belief densities, $N(p_1; \mu_1, \sigma_1)$ and $N(p_2; \mu_2, \sigma_2)$ over their volatile parameters, where $p_1, p_2 \in [0, 1]$ are the probabilities (i.e. WS availabilities), $\mu_1, \mu_2 \in [0, 1]$ are the means, and $\sigma_1, \sigma_2 \in \mathbb{R}$ are the standard deviations of the w_1 and w_2 densities, respectively. Using Theorem 7 on pg. 141 of Rohatgi [12], we derive the general belief densities of the composite WS containing both services for four commonly used configurations:

- **Sequential flow** Because each of the services w_1 and w_2 in a sequential flow are executed, the composite beliefs depend on both $N(p_1; \mu_1, \sigma_1)$ and $N(p_2; \mu_2, \sigma_2)$. If executions of the WSs are independent of each other, then the probability of composite service c containing WSs w_1 and w_2 in sequence, p_c , is the product of the individual WS probabilities: $p_c = \prod_{i=1}^2 p_i$. Note that because p_c is a product of the individual probabilities, we could view the range of p_c (i.e. $[0,1]$) as a prod-

uct space. The resulting density, $F(p_c)$, may then be found as follows:

$$F(p_c) = \int_0^1 N(p_1; \mu_1, \sigma_1) N\left(\frac{p_c}{p_1}; \mu_2, \sigma_2\right) \frac{1}{|p_1|} dp_1 \quad (2)$$

To simplify Eq. 2 we introduce the operator \otimes in the following way:

$$F(p_c) = N(p_1; \mu_1, \sigma_1) \otimes N(p_2; \mu_2, \sigma_2) \quad (3)$$

- **Concurrent flow** Analogous to the sequential flow, each of the services in a parallel flow must also be executed. Hence, $F(p_c)$ is derived analogously to the one for the sequential flow.
- **Conditional flow** Any one of several branches is executed in a conditional flow. For example, let there be two branches that are followed with probabilities, p' and p'' and $(p' + p'' = 1)$ ³. p_c is then the weighted sum of the probabilities of the individual WSs: $(p_c = p' \times p_1 + p'' \times p_2)$. Given the beliefs over the probabilities p_1 and p_2 , we may derive the belief density over p_c as:

$$F(p_c) = \frac{1}{p'p''} \int_0^1 N\left(\frac{p_1}{p'}; \mu_1, \sigma_1\right) N\left(\frac{p_c - p_1}{p''}; \mu_2, \sigma_2\right) dp_1 \quad (4)$$

- **Loop** For simplicity, we restrict our analysis to loops that are iterated a fixed number of times, say n ⁴. Analogous to the sequential flow, each WS in a loop must be executed n times. Assuming both w_1 and w_2 are both contained in the loop, we may take the product $p'_c = p_1 \times p_2$ n times:

$$F(p_c) = (F(p'_c) \otimes F(p'_c)) \otimes F(p'_c) \dots n \text{ times} \quad (5)$$

³These probabilities may be obtained from prior interactions.
⁴This precludes loops that conditionally terminate (e.g. while loops). The difficulty is in knowing the number of times the loop will run a priori.

We may now derive the belief density for a WSC using any combination of the four constructs by using the methods described above.

Sampling Algorithm Although we model the densities over availability of individual WSs as Gaussians (for example Figs. 2(a) and (b)), the function over the product space is not a Gaussian but rather a modified Bessel function of the second kind [7]. 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. 3. We first sample the individual densities and tabulate the frequencies of the product of the two independent variables (the assignment to p in the algorithm). If the flow pattern used in the WSC is a conditional flow or loop, we adjust the assignment of p as we have described above. The frequency histogram is then 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. 2(c). Densities analogous to this one are summed to obtain the broker’s belief over the volatility of the aggregate parameter of the composite WS Insurance Information.

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 $s_1 \sim \mathcal{N}(\mu_1, \sigma_1), s_2 \sim \mathcal{N}(\mu_2, \sigma_2)$
 $p \leftarrow s_1 \times s_2$
 $j \leftarrow$ bin corresponding to p
 Increment $frequencyCountBins[j]$ by 1
end for
Convert $frequencyCountBins$ histogram to a normalized cdf
Convert the resulting cdf to a pdf
end algorithm

Figure 3. Sampling algorithm for approximating a probability density over a product of two independent random variables with Gaussian distributions.

4.2 Algorithm

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:

$$a^* = \underset{a \in A}{\operatorname{argmax}} \operatorname{VOC}_{T(\cdot|a,s^*)}(s) \quad (6)$$

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

1. $s \leftarrow s_0$
2. $n \leftarrow H$
3. **while** $n > 0$
4. **if** $\operatorname{VOC}^*(s) > \operatorname{QueryCost}(T'(\cdot|a^*, s'))$
5. **if** a^* is composite
6. $a_k^* \leftarrow \operatorname{findWS}^*(l-1, A, s)$
7. Query a_k^* //primitive WS
8. $T' \leftarrow \operatorname{UpdateModel}(\text{Level } l, \text{Level } k)$
9. Calculate policy π_l^* using the new MDP with T'
10. $a \leftarrow \pi_l^*(s)$
11. **if** a is composite
12. Recursively call **AWSC** for a at level $l-1$ and π_{l-1}^*
13. **else**
14. Execute primitive Web service a
15. Get response of a and construct next state s'
16. $s \leftarrow s'$
17. $n \leftarrow n-1$
18. **end while**

Figure 4. Algorithm for executing and adapting a hierarchical WSC to revised information.

The algorithm for an adaptive WSC is shown in Fig. 4. 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. 5.

Algorithm for findWS* (k, A, s^k)
Input: k //level, A //action set, s^k //state at level k

1. $a_k^* \leftarrow \underset{a \in A}{\operatorname{argmax}} \operatorname{VOC}_{T(\cdot|a,s^*)}(s^k)$
2. **if** a_k^* is a primitive WS **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** $\operatorname{findWS}^*(k-1, A, s^{k-1})$

Figure 5. Function findWS* recursively finds a WS that yields the highest VOC.

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. 6.

The following theorem describes the complexity of the above algorithms. We do not show the proof here due to space limitations, but refer the reader to [8] for the proof.

Theorem 1 Let N denote the number of possible values of

Algorithm for UpdateModel(l, k)
Input : k //depth, l //current level

1. **if** $l > k$ **then**
2. $T' \leftarrow \text{UpdateModel}(l - 1, k)$
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'

Figure 6. Function *UpdateModel* recursively updates the transition probabilities and policies in the hierarchical WSC using the revised information.

a random variable X_i in the state space, H denote the horizon, $|A|$ denote the largest number of services at any level of the hierarchy. The worst-case complexity of the algorithm in Fig. 4, is:

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

Theorem 1 reinforces the intuition that the complexity of the hierarchical WSC algorithm will grow exponentially as the number of levels increases.

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 different querying strategies.

5.1 Architecture

The algorithm in Fig. 4 is implemented as a WS-BPEL flow and all WSs were implemented using WSDL. To the WS-BPEL flow, we give the optimal policies, $\langle \pi_l^*, \pi_{l-1}^*, \dots, \pi_0^* \rangle$, the start state, and horizon as input. We only need to make a small change to the SOA introduced in [9] to accommodate hierarchical WSCs.

Within our SOA, we provide internal WSs that generate the policy for the MDP model and for computing the VOC. If the VOC exceeds the cost of querying a particular service, the WS-BPEL flow invokes a special WS whose function is to query the service for revised information. This information is used to get a new MDP policy to be fed back to the WS-BPEL flow. The 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, continuing until the goal state is reached or the total number of steps are 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).⁵ We model the mortgage broker’s beliefs over the volatile parameter of the individual WSs, $(Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 1) using *Gaussian* density functions. For composite WSs, we derive the densities over the aggregated parameters as shown in Section 4.1.

In Fig. 7(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. 2). We ensured that the compositions using the different strategies received similar responses from the services.

For comparison, we used four other heuristics for querying selectively. The first is the use of a *static policy*, where adaptation is ignored. The second is a *random query* approach, which queries a random service at every time step in the execution. The third employs an *intermittent query* approach, which, unlike the previous approach, queries randomly at alternate stages of the composition. The fourth is the *largest difference* approach, which queries the service 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 simple analogies of certain aspects of the VOC based approach, making them an effective comparison testbed.

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 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

⁵Of course, the rate of request satisfaction, for example, would depend on loan amount and other factors; we assume that these will be provided.

Mortgage Loan Acquisition Process

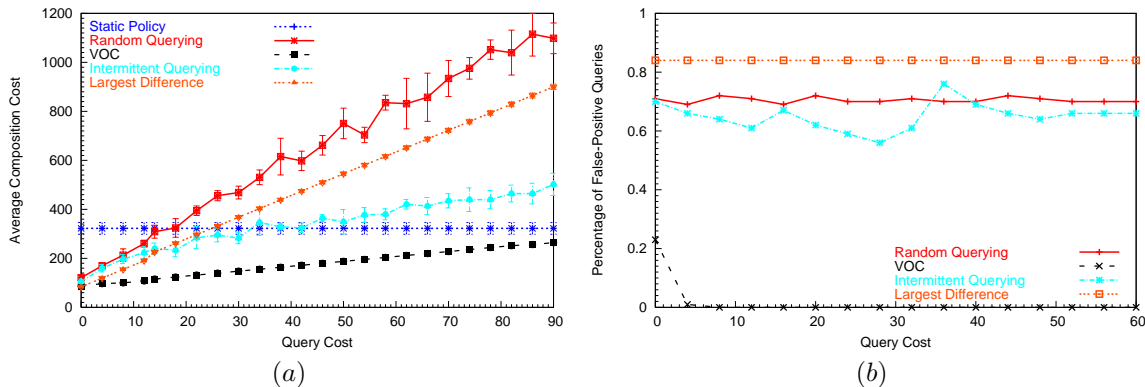


Figure 7. Comparisons of the VOC based adaptive WSC with the static policy and other querying approaches for the mortgage loan acquisition. (a) We measure the average cost. Lower cost indicates better performance. (b) We measure the percentage of false-positive queries. Notice that VOC results in a WSC 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.

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 summary, a WSC that is adapted using VOC 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. 7(b), VOC based querying results in significantly less false positives compared to other strategies. The number drops to zero as the query cost increases because a query is issued only if the VOC exceeds the cost of querying. 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 Fig. 8, we give the run times for each of the comparative approaches. To be realistic, we included a short lag time (5 ms) in receiving the query response from the WSCs. We first observe that an adaptive WSC that uses VOC runs two to three times slower than a WSC that does not adapt such as the static policy. However, the difference is less when other strategies are utilized. The additional runtime of the VOC is primarily due to the computations required for Eq. 1. Notice, however, that as the query cost increases, the time required by the VOC based approach de-

Query strategies	Time (ms)		
	QC = 0	QC = 40	QC = 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	653 ± 4	540 ± 20	470 ± 20
Largest difference	535 ± 8	535 ± 12	541 ± 11

Figure 8. Running times of the various querying strategies for different query costs. We used Linux platform on Xeon 3.8GHz with 2.0GB memory.

creases, because it issues less queries.

6 Related Work

In the context of data volatility, Chafle et al. [4] use a staged composition approach for adaptation by pre-specifying multiple alternate plans at the logical, physical, and runtime levels of a WSC. While capable of adapting to different events, many of the alternative plans may not be used, making the approach inefficient. Paques et al. [10] 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. [6] use a technique that manages the dynamism of WSC environments through Bayesian learning by updating parameters based on previous interactions with the individual WSCs. This method suffers from being slow in param-

ter updates, and may result in WSC recomputations that do not bring change. Au et al. [1] introduce a framework that composes WSCs in the presence of data volatility. Using a reactive querying policy, they obtain current parameters of WS providers when the parameters expire. Plan recomputation takes place regardless of whether the revised parameter values are expected to change the composition.

The composition of hierarchical WSCs is also a relatively new topic in the SOA literature. WSC tools such as SHOP2 [13] and HALEY [14] have been introduced to accommodate the hierarchies in WSCs. Both, however, focus on composition and assume the parameters used to model the environment remain static throughout execution.

7 Conclusion

Real world process environments are volatile – non-functional parameters of the services may vary over time. In such environments, WSCs must adapt to the revised information to remain cost effective. VOC has been proposed to gauge the value of the expected change that revised information may bring to the WSC. While previously, VOC was applied toward adapting simple flat WSCs, in this paper, we extended its applicability to hierarchical WSCs. This is significant because real-world 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.

References

- [1] T.-C. Au and D. Nau. Reactive query policies: A formalism for planning with volatile external information. In *CIDM*, pages 243–250, 2007.
- [2] A. Borgida and T. Murata. Tolerating exceptions in workflows: a unified framework for data and processes. In *WACC*, pages 59–68, 1999.
- [3] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004.
- [4] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava. Adaptation in web service composition and execution. In *ICWS, Industry Track*, 2006.
- [5] N. Desai, A. K. Chopra, and M. P. Singh. Business process adaptations via protocols. In *SCC'06*, pages 601–608, 2006.
- [6] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using markov decision processes. *JWSR*, 2(1):1–17, 2005.
- [7] A. G. Glen, L. Leemis, and J. H. Drew. Computing the distribution of the product of two continuous random variables. *CSDA*, 44(3):451–464, 2004.
- [8] J. Harney and P. Doshi. Selective querying using value of changed information for web service compositions. Technical report, December 2008. <http://www.cs.uga.edu/jfh/research/TR12-08.pdf>.
- [9] J. Harney and P. Doshi. Selective querying for adapting web service compositions using the value of changed information. *IEEE TSC*, 1(3):169–185, July–September 2009.
- [10] H. Paques, L. Liu, and C. Pu. Adaptation space: A design framework for adaptive web services. *JWSR*, 1(3):1–24, 2004.
- [11] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, NY, 1994.
- [12] V. Rohatgi. *An Introduction to Probability Theory and Mathematical Statistics*. John Wiley & Sons, NY, 1976.
- [13] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating daml-s web services composition using shop2. In *ISWC*, 2003.
- [14] H. Zhao and P. Doshi. Haley: A hierarchical framework for logical composition of web services. In *ICWS*, pages 312–319, 2007.