Adaptive Web Processes Using Value of Changed Information

John Harney and Prashant Doshi

LSDIS Lab, Dept. of Computer Science,
University of Georgia, Athens, GA 30602
{jfh, pdoshi}@cs.uga.edu

Abstract. Web process composition is receiving much attention as an important problem for the services oriented computing community. Most compositions built by planning methods use a pre-defined model of the process environment. These methods have assumed that the information in the models and consequently the compositions remain static and accurate throughout the life cycle of the Web process. We describe an approach that accounts for the dynamic nature of services by formulating a system that queries external sources intelligently. We give a method for measuring the value of change that revised information may potentially introduce in the Web process. We provide an algorithm that calculates and uses this value to optimally adapt the Web process to possible changes in the environment. Using two realistic scenarios, we show our idea and compare its performance to alternative approaches.

1 Introduction

Planning based approaches to Web process composition [1, 2, 3] rely on pre-specified models of the process environment to generate plans. For example, decision-theoretic planners such as Markov decision processes (MDPs) [2, 4] utilize a model, which describes the state-action transition probabilities and the costs of service invocations, to generate a policy that guides the composition. The optimality of the Web process is dependent on the accuracy with which the model captures the process environment. In volatile environments [5] where the characteristics of the process participants may change frequently, the Web process may become suboptimal if it is not updated with the changes. As an example, consider a supply chain scenario in which a manufacturer has the option of ordering goods from either its preferred or another supplier. The ordering of the manufacturer's actions depends on the probability with which the preferred supplier usually satisfies the orders and the cost of using the preferred supplier. If the preferred supplier's rate of order satisfaction drops suddenly (due to unforeseen circumstances), and the manufacturer does not revise its model to reflect this change, its Web process will continue to utilize that supplier over others.

A straightforward approach to address this problem is to query the model parameters periodically, update them if they have changed, and reformulate the

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 179–190, 2006.

Web process based on the updated model. [1] This approach does not account for the cost of querying the model parameters, which may be more expensive than using a suboptimal Web process. For example, finding out a supplier's current rate of order satisfaction may be more expensive than a reduction in expected cost that the new information will entail for the Web process.

In this paper, we introduce a method to intelligently adapt the Web process to volatile environments by computing the *value of changed information (VOC)*. In this method, we compute the tradeoff between the cost of querying for revised information and the expected value of the change in the Web process that the revised information will bring. We update the model parameters and compose the Web process again, only if the VOC is greater than the query cost. We adopt a *myopic* approach in that we query only one service provider at a time and utilize the revised information from that provider which leads to the maximum VOC. We show that, though myopic, the approach performs reasonably well in adapting the Web process to the changes in the environment. In particular, our experiments demonstrate that the VOC mechanism avoids "unnecessary" queries in comparison to the naive approach of periodic querying. This translates to a savings in overall costs for the Web process. For the purpose of evaluation, we utilize two scenarios - a supply chain and a clinical administrative pathway. Within our services-oriented architecture (SOA), we represent the manufacturer's and hospital's Web processes using WS-BPEL [6], and the provider services as well as a service for computing the VOC using WSDL [7].

We point out that the VOC computation shares its conceptual underpinnings with the value of perfect information (VPI) [8]. This is due to the fact that they are both special cases of the value of information idea, which attempts to determine if new information is indeed 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. Both quantities cannot be negative. The updated information may lead to a Web process whose total cost is greater than before. For example, if the preferred supplier's probability of meeting orders drops considerably, the manufacturer may be forced to drop the preferred supplier in favor of using a more expensive supplier. Nevertheless, as we show, the revised Web process incurs less total cost in the changed environment in comparison to the original Web process in this environment.

2 Related Work

Only recently have researchers turned their attention to managing processes in volatile environments. Au et al. [5] obtains current parameters about the Web process by querying specific Web service providers when the values expire. Plan recomputation is assumed to take place irrespective of whether the revised parameter values are expected to bring about a change in the composition. This

[1] In general, new information may require a complete recomposition of the Web process to remain optimal, though sometimes only local changes may be sufficient.

may lead to frequent unnecessary computations. Muller et al. [9] propose a workflow adaptation strategy based on pre-defined event-condition-action rules that are triggered when a change in the evironment occurs. While the rules provide a good basis for performing contingency actions, they are limited in the fact that they cannot account for all possible actions and scenarios that may arise in complex workflows. Additionally, the above works do not address long term optimality of process adaptation. [2] offers such a solution using a technique that manages the dynamism of Web process environments through Bayesian learning. The process model parameters are updated based on previous interactions with the individual Web services 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 recomputations that do not bring about any change in the Web process.

3 Motivating Scenarios

In order to illustrate our approach we present two example scenarios:

Supply Chain. A manufacturer receives an order to deliver some merchandise to a retailer. The manufacturer may satisfy the order in one of several ways. He may satisfy the order from his own inventory if sufficient stock exists. The manufacturer may request the required parts from a preferred supplier. The manufacturer may also search for a new supplier of parts, or buy them on the spot market. A *costing analysis* reveals that the manufacturer will incur least cost if he is able to satisfy the order from his own inventory. The manufacturer will incur increasing costs as he tries to fulfill the order by procuring parts from his preferred supplier, a new supplier, and the spot market.

In Fig. 1, we depict the supply chain scenario. The manufacturer may choose from several processes. For example, the manufacturer may attempt to satisfy

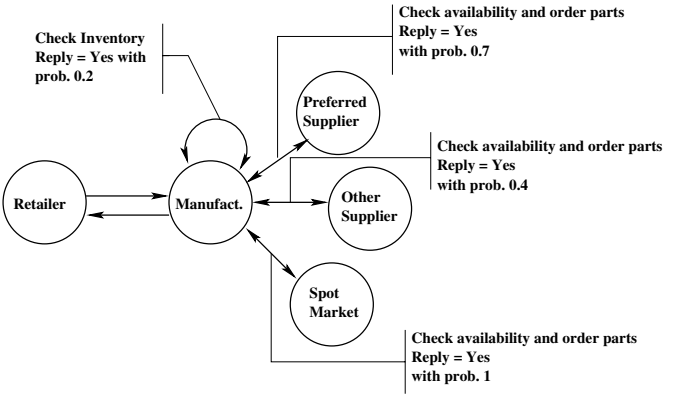


Fig. 1. Collaboration diagram showing interactions between the business partners in our motivating scenario. We have used example probability values to aid understanding.

the order from his inventory. If unable to do so, he may resort to order from the preferred supplier. Another process may involve bypassing the inventory check, since the manufacturer strongly believes that his inventory will not satisfy the order. He may then initiate a status check on his preferred supplier. These example processes reveal two important factors for selecting the optimal one. First, the manufacturer must accurately know the certainty with which his order will be satisfied by the various suppliers. Second, rather than selecting an action with the least cost at each stage, the manufacturer must select the action expected to be optimal over the long term.

Patient Transfer. A hospital receives a patient who has complained of a particular ailment. The patient is first checked into the hospital and then seen by one of the hospital's physicians. He may, upon examination, decide to transfer the patient to a secondary care provider for specialist treatment. We assume that the hospital has a choice of four secondary care givers to select from with differing vacancy rates and costs of treatment, with the preferred one having the best vacancy rate and least cost (see Fig. 2).

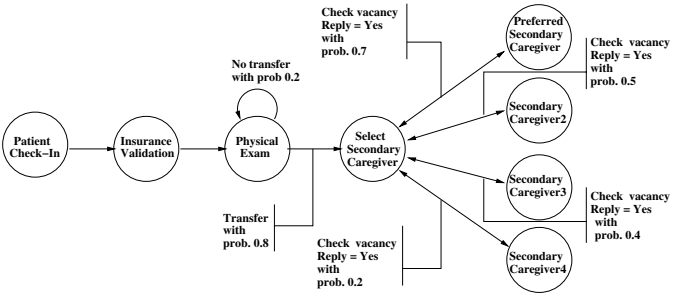


Fig. 2. The patient transfer clinical pathway for a primary caregiver. As before, we have used example probability values to aid understanding.

Similar to our previous example, several candidate Web processes present themselves. For example, the physician may decide not to transfer the patient, instead opting for in-house treatment. However, if the physician concludes that specialist treatment is required, several factors weigh in toward selecting the secondary care giver. These include, the typical vacancy rates, costs of treatment, and geographic proximity of the caregivers.

4 Background: Web Process Composition Using MDPs

As we mentioned before, our approach is applicable to any model based process composition technique. For the purpose of illustration, we select a decision-theoretic planning technique for composing Web processes [2]. Decision-theoretic planners such as MDPs model the process environment, WP, using a sextuplet:

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

where $S = \Pi_{i=1}^{n} X^i$, where S is the set of all possible states factored into a set, X, of n variables, $X = \{X^1, X^2, \ldots, X^n\}$; A is the set of all possible actions; T is a transition function, $T : S \times A \rightarrow \Delta(S)$, which specifies the probability measure over the next state 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 supply chain scenario as a MDP. The state of the workflow is captured by the random variables – **Inventory Availability**, **Preferred Supplier Availability**, **New Supplier Availability**, **Spot Market Availability**, **Order Assembled**, and **Order Shipped**. A state is then a conjunction of assignments of either *Yes*, *No*, or *Unknown* to each random variable. Actions are Web service invocations, A={Check Inventory Status, Check Preferred Supplier Status, Check New Supplier Status, Check Spot Market Status, Assemble Order, Ship Order}. The transition function, T, models the non-deterministic effect of each action on some random variable(s). For example, invoking the Web service Check Inventory Status will cause **Inventory Availability** to be assigned *Yes* with a probability of T(**Inventory Availability**=*Yes*|Check Inventory Status, **Inventory Availability**=*Unknown*), and assigned *No* or *Unknown* with a probability of (1-T(**Inventory Availability**=*Yes*|Check Inventory Status,**Inventory Availability**=*Unknown*)). The cost function, C, prescribes the cost of performing each action. This includes the financial as well as the infrastructural cost to the manufacturer of using a service. We let H be some finite value which implies that the manufacturer is concerned with getting the most optimal Web process 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 our manufacturer has modeled its Web process composition problem as a MDP, he may apply standard MDP solution techniques to arrive at an optimal process. These solution techniques revolve around the use of stochastic dynamic programming [4] for calculation of the optimal policy using *value iteration*:

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

where:

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

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 minimum 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.

$$\pi^*(s) = \underset{a \in A}{argmin} \; Q^n(s, a) \tag{3}$$

In Eq. 3, π^* is the optimal policy which is simply a mapping from states to actions, $\pi^* : S \rightarrow A$. The Web process is composed by performing the WS invocation prescribed by the policy given the state of the process and observing the results of the actions. Details of the algorithm for translating the policy to the Web process are given in [2].

5 Value of Changed Information

Several characteristics of the process participants – service providers – may change during the life-cycle of a Web process. For example, the cost of using the preferred supplier's services may increase, and/or the probability with which the preferred supplier meets the orders 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 paper, we focus on a change in the transition function T, though our approach is generalizable to fluctuations in other model parameters too.

Not all updates to the model parameters cause changes in the process composition. 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.

5.1 Definition

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. In the supply chain example, this would translate to asking, say, only the preferred supplier for its current rates of order satisfaction, as opposed to both the preferred supplier and the other supplier, simultaneously. The revised information may change the following transition probabilities, T(**Preferred Supplier Availability** = Yes | Check Preferred Supplier Status, **Preferred Supplier Availability** = Unknown), and T(**Preferred Supplier Availability** = No | Check Preferred Supplier Status, **Preferred Supplier Availability** = 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 current belief distributions over their 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,

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

where $T'(\cdot|a, s')$ represents the distribution that may be queried and subsequently may get revised, $\mathbf{p} = \langle p_1, p_2, \ldots, p_n \rangle$ represents a possible response to

the query (revised distribution), n is the number of values that the variable under question may assume, and $Pr(\cdot)$ is our current *belief* over the possible values. As a simple illustration, let us suppose that we intend to query the preferred supplier for its current rate of order satisfaction. Eq. 4 becomes,
$EV(s) = \int_{\langle p,1-p\rangle} Pr(T'$ (**Pref. Supp. Avail.**=Yes/No |Check Pref. Supp. Status, **Pref. Supp. Avail.** = Unknown)= $\langle p, 1-p\rangle)$ $V_{\pi^*}(s|T')dp$
assuming that the random variable **Preferred Supplier Availability** assumes either *Yes* or *No* on checking the status of the preferred supplier.

Let $V_\pi(s|T')$ be the expected cost of following the original policy, π from the state s in the context of the revised model parameter, T'. We recall that the policy, π, is optimal in the absence of any revised information. We formulate the value of change due to the revised transition probabilities as:

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

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'.

Analogous to the value of perfect information, the following proposition holds for VOC.

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

Proof. The proposition follows trivially if we find that $\forall s, \mathbf{p}$ $V_\pi(s|T') - V_{\pi^*}(s|T') \geq 0$. By definition (Eq. 3), π^* is an optimal policy for the revised model. This implies that for any other policy, $\pi' \in \Pi \backslash \pi^*$, where Π is the space of all policies, $\forall \mathbf{p} \quad V_{\pi'}(s|T') \geq V_{\pi^*}(s|T')$. This holds true over all the states. The required inequality obtains since π must either be in $\Pi \backslash \pi^*$, or be equal to π^*. □

Since querying the model parameters and obtaining the revised information may be expensive, we must undertake the querying only if we expect it to pay off. In other words, we query for new information from a state of the Web process 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.

5.2 Web Process Composition with VOC

In order to formulate and execute the Web process, we simply look up the current state of the Web process in the policy and execute the WS prescribed by the policy for that state. The response of the WS invocation determines the next state of the Web process. We adapt the composition of the Web process

```
Algorithm for adaptive Web process
    Input: π* //optimal policy, s_0 //initial state
    s ← s_0
    while goal state not reached
        if VOC*(s) > QueryCost(T'(·|a, s')
            Query service provider, a (Eq. 6), for new probabilities
            Form the new transition function, T'
            Calculate policy π* using the new MDP model
            with T'
        a ← π*(s)
        Execute the Web service a
        Get response of a and construct next state, s'
        s ← s'
    end while
end algorithm
```

Fig. 3. Algorithm for adapting a Web process to revised information and executing it

to fluctuations in the model parameters, by interleaving the formulation with VOC computations. The algorithm for the adaptive Web process composition is shown in Fig. 3.

For each state encountered during the execution of the Web process, we query a service provider for new information if the query is expected to bring about a change in the Web process that exceeds the query cost. For example, in the supply chain process, we select and query a supplier for its current rate of order satisfactions. Of all the suppliers, we select the one whose possible new rate of order satisfaction is expected to bring about the most change in the Web process, and this change exceeds the cost of querying that provider. In other words, we select the service provider associated with the WS invocation, a, to possibly query for whom the VOC is maximum:

$$a = \underset{a \in A}{argmax} \; VOC_{T(\cdot|a,s')}(s) \tag{6}$$

Let VOC*(s) represent the corresponding maximum VOC.

Our algorithm does not consider the cost of VOC computation in deciding whether to query a service provider. In particular, this would require knowing what the possible VOC computation cost could be, and a way to compare computation cost with WS invocation cost using inter-convertible units. We avoid these complications under the assumption that sufficient inexpensive computational resources are available to perform VOC computations.

6 Experiments

We first outline our SOA, in which we wrap the VOC computations in WSDL based internal Web services, followed by our experimental results on the performance of the adaptive Web process.

6.1 Architecture

The algorithm described in Fig. 3 is implemented as a WS-BPEL flow while all WSs were implemented using WSDL. To the WS-BPEL flow, we gave the optimal policy, π^*, and the start state as input. Our experiments utilized IBM's BPWS4J engine for executing the BPEL process. We show our SOA in Fig. 4.

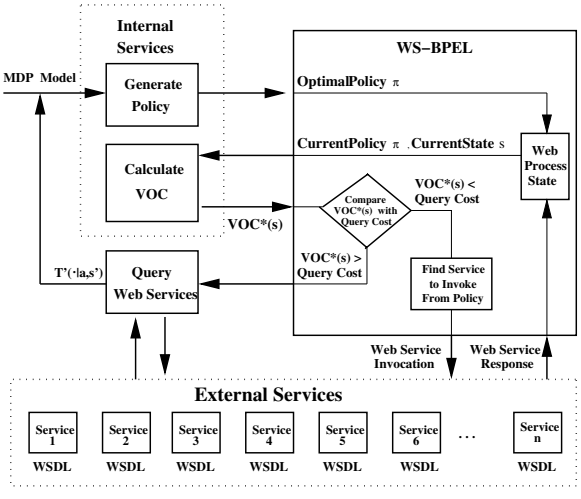


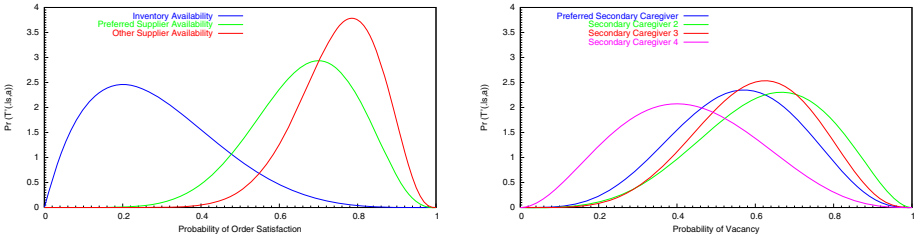
Fig. 4. SOA for implementing our adaptive Web process

Within our SOA, we provide internal WSs for solving the MDP and generating the policy, and computing the VOC. If the $VOC^*(s)$ 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 WS 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 and the response is used to formulate the next state of the process. This procedure continues until the goal state is reached.

6.2 Performance Evaluation

We experimentally compare the performance of our VOC based approach for adapting a Web process with two other methods, for both the supply chain and the patient transfer examples outlined in Section 3. The first method assumes that there is no adaptation to the volatile environment and uses the same policy for every execution of the process. A MDP model is formulated and solved before the first execution instance and the resulting policy is used for every instance then onwards. The second method implements a periodic querying strategy, in which a service provider is selected at random and queried for revised information

at each state of the Web process. Using the new information, the policy is resolved and the Web process continues to run using the new policy. For the supply chain example, we queried the inventory or the suppliers for their current percentage of order satisfaction [2], while in the patient transfer pathway, we queried the secondary caregivers for their current vacancy rates. [3]



(a) (b)

Fig. 5. The probability density functions representing (*a*) the manufacturer's belief over the suppliers' rates of satisfaction in the supply chain scenario; (*b*) the primary caregiver's beliefs over the secondary caregivers' probabilities of having a vacancy

We model the manufacturer and primary caregiver's beliefs over the possible parameters of the service providers, ($Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 5) using *beta* density functions. Other density functions such as Gaussians or polynomials may also be used. Fig. 5(*a*) shows the beta densities that represent the manufacturer's distribution over the rate of order satisfaction by the inventory ie. $T'(\textbf{Inv. Avail.} = Yes|\textsf{Check Inv.}, \textbf{Inv. Avail.} = Unknown)$, and analogously for the preferred and other suppliers. Means of the densities reveal that the inventory tends to be less reliable in satisfying orders than other suppliers. Fig. 5(*b*) shows the density plots over the probability of a vacancy with the preferred and other secondary caregivers.

In Fig. 6 we compare the three strategies with respect to the average cost incurred from the execution of the Web process, as the cost of querying the service providers is increased. Our methodology consisted of running 100 independent instances of each process within a simulated volatile environment, where the queried parameters of the service providers were distributed according to the density plots in Fig. 5. We ensured that the processes using each of the three strategies received similar responses from the service providers.

Intuitively, as we increase the cost of querying, our VOC based approach performs less queries and adapts the Web process less. For large query costs, its performance is similar to using a Web process with a static policy strategy. In Fig. 6(*a*) and (*b*), we show the results for the supply chain and patient transfer

[2] In the real world, an example response by a supplier could be, "We are currently meeting 2 of every 3 orders".

[3] Of course, the rate of order satisfaction would depend on the quantity of the order and other factors; we assume that these will be provided to the suppliers.

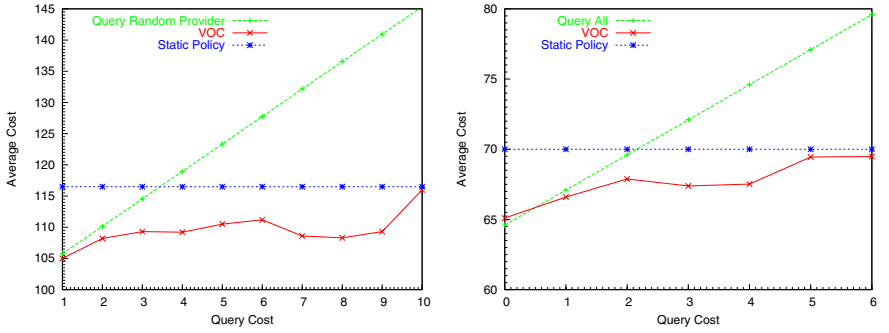


Fig. 6. Comparisons of the VOC based process composition with the static policy and periodic querying approaches for (a) supply chain, and (b) patient transfer scenarios

scenarios respectively. For smaller query costs, a VOC based approach will query frequently, though not as much as a strategy that always queries a provider. As we increase the query costs, the VOC based approach will allow a query for revised information only if its value exceeds the cost. Thus, a Web process that is adapted using VOC performs better (incurs less average cost) than periodic querying in a volatile environment because only significant changes to the Web process are carried out while simultaneously avoiding frequent costly queries.

We point out that the improvement in overall costs comes at a computational price, as illustrated by our execution time results in Table 1. Calculating the VOC as shown in Eq. 5 is computationally intensive. The probability $\mathbf{p}$ represents a revised probability of transition on performing an action. To calculate the VOC, we must compute $V_{\pi}(s|T')$ and $V_{\pi^*}(s|T')$ for all possible $\mathbf{p}$ and average over their difference based on our distribution over $\mathbf{p}$ (shown in Fig. 5). The revised value function $V_{\pi^*}(s|T')$ must be computed using the standard value iteration defined in Eq. 1. The integral in Eq. 5 is approximated using monte carlo sampling, which provides a faster evaluation technique and contains negligible error. Exploring additional efficient calculations of VOC is one avenue of future work.

Table 1. Execution times of Web processes using the VOC and query always approaches for the supply chain and patient transfer scenario

Problem	VOC	Query always
Supply Chain	1.43s	0.37s
Patient Transfer	18.6s	7.4s

Our experiments provide two conclusions: First, by augmenting Web process composition with VOC calculations, significant information changes in volatile environments are considered and used to make better decisions about which services to invoke next. The comparison of VOC and static policy implementations

illustrate that the overall average cost of the Web process using VOC is significantly less than utilizing a non-changing policy. Second, we demonstrated that an intelligent strategy of obtaining revised information that accounts for the cost in obtaining the information results in less expensive Web processes than a naive method of periodic querying for new information.

7 Discussion

Real-world process environments are volatile–parameters of the service providers ranging from costs to reliability may change over time. In such environments, Web process compositions must adapt to the revised information to remain cost-effective. We presented a method that intelligently adapts a Web process to changes in parameters of service providers. Specifically, our approach measures the expected value of change that the revised information may bring to Web processes and compares it with the cost of obtaining the information. If the revised information is worth the cost of obtaining it, we query the providers for their current parameters and reformulate the Web process using the revised information. Using two example scenarios, we show that our approach results in Web processes that are more cost-effective than approaches that do not change the composition or use a simple periodic querying strategy. Our future line of work will involve attempting to compute the VOC more efficiently, and understand the trade off between VOC calculation accuracy and computational efficiency.

Acknowledgements. This research was supported by a grant from UGARF.

References

[1] Srivastava, B., Koehler, J.: Planning with workflows - an emerging paradigm for web service composition. (2004)
[2] Doshi, P., Goodwin, R., Akkiraju, R., Verma, K.: Dynamic workflow composition using markov decision processes. J of Web Services Research **2**(1) (2005) 1–17
[3] Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D.: Automating daml-s web services composition using shop2. In: International Semantic Web Conference. (2003)
[4] Puterman, M.L.: Markov Decision Processes. John Wiley & Sons, NY (1994)
[5] Au, T.C., Kuter, U., Nau, D.S.: Web service composition with volatile information. In: International Semantic Web Conference. (2005) 52–66
[6] IBM: Business Process Execution Language for Web Services version 1.1. (2005)
[7] W3C: Web Services Description Language (WSDL) 1.1. (2001)
[8] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (Second Edition). Prentice Hall (2003)
[9] Muller, R., Greiner, U., Rahm, E.: Agentwork: a workflow system supporting rule-based workflow adaptation. J of Data and Knowledge Engg. **51**(2) (2004) 223–256