

Selective Querying For Adapting Web Service Compositions Using the Value of Changed Information

John Harney and Prashant Doshi

Abstract—Web service composition (WSC) techniques assume that the parameters used to model the environment remain static and accurate throughout the composition’s execution. However, WSCs often operate in environments where the parameters of its component services are volatile. To remain optimal, WSCs must adapt to these changes. Adaptation requires up-to-date knowledge about the revised parameters of each of the services. One way of obtaining this knowledge is to query services for their revised parameters. Querying services for their parameters could be time-consuming and expensive. We must therefore carefully manage how queries are conducted. Specifically, an adaptive WSC must know when to query for revised information, and from which service(s) to obtain information. We present a method to selectively query services using the value of changed information (VOC). VOC measures the value of the change that revised information may potentially introduce to the composition. We reduce the complexity of computing the VOC, first by anticipating values of the service parameters that do not change the WSC, and second by using parameter expiration times obtained from predefined service-level agreements. Using two scenarios, we illustrate our approach and demonstrate the computational savings theoretically and experimentally.

Index Terms—Optimization of Services Composition, Value of Information, Modeling

I. INTRODUCTION

Web service composition (WSC) techniques assume that the parameters used to model the environment remain static and accurate throughout the composition’s execution. The compositions are built using a pre-defined model of the environment obtained at design time and executed. This fundamental assumption is unrealistic as environments may change during execution. For example, a product may go out of stock affecting its availability, the network bandwidth may fluctuate affecting the WS response time, or costs of using a travel agent’s service may increase. Many techniques do not adapt compositions to changes, thereby producing sub-optimal results.

Dynamism manifests in WSC environments in a variety of ways. For example, changes range from the operational level (such as a newly introduced task) to the organizational level (such as new company policies) as mentioned in [3], [4]. Indeed, the surveys classify a variety of changes in many different ways. Solutions have been presented to address some

these changes, ranging from exception-handling techniques defined in [5] to instituting protocol adaptations defined in [6].

However, less attention has been paid to *data volatility* that exists during execution. As a concrete example, consider a supply chain in which two suppliers compete for orders from a large manufacturer. The sequence in which the manufacturer utilizes the services of the two suppliers would depend on the probability with which the suppliers usually satisfy the orders and the cost of using them. If the preferred supplier’s rate of order satisfaction drops suddenly (due to say, a warehouse fire), a cost-conscious manufacturer should replace it with another supplier to remain optimal. Important service parameters such as cost, availability, or the rate of order satisfaction 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 composition¹.

Thus, the WSC must possess up-to-date knowledge of the revised information during execution. To obtain this knowledge, an adaptive WSC may query component services – typically their providers – for their revised parameter values. The changed 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 on 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 Internet). As a result, querying for information from service resources is often tedious, time consuming, and costly. Queries must therefore be carefully managed – we should only query those services whose parameter changes may potentially impact the WSC so as to minimize the additional 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 this article, we present approaches that address these concerns. We introduce a method to intelligently query using the *value of changed information (VOC)*. In particular, we compute the trade-off between the cost of querying for up-to-date information² and the value of expected change in

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

²Often, contractual agreements are required to obtain the revised parameters from the service providers. Nevertheless, we also consider the case that the revised information may be obtained at no cost.

the WSC that the revised information will bring. We update the model parameters and compose the WSC 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.

As computing the VOC is expensive, we present two techniques that alleviate the complexity of computing it. First, we observe that for particular values of the parameters, the WSC remains unchanged. Consequently, such revised values may be ignored, as they do not change the WSC. We provide a simple and quick way to ascertain these values. Second, we may utilize the expiration times often associated by service providers to the parameters of their services. We use the intuition that we need not consider querying those services (or associated information providers) for revised information whose previously obtained information has not expired. We incorporate this insight into the VOC formulation, and call the approach, the *value of changed information with expiration times* (VOC^E). Because VOC^E focuses the computations on only those services whose parameters could have changed, it is computationally more efficient than the traditional VOC.

We show that, though myopic, the approach performs well in the presence of service data volatility, resulting in reduced time and cost overhead caused by querying. In particular, our experiments demonstrate that the VOC mechanism avoids “unnecessary” queries in comparison to a naive approach of querying periodically and other heuristic approaches. This translates to savings in overall costs for the WSC.

For the purpose of evaluation, we utilize two realistic scenarios – a supply chain and a clinical administrative pathway. Within our services-oriented architecture (SOA), we represent the manufacturer’s and hospital’s WSCs using WS-BPEL, and the provider services as well as a service for computing the VOC using WSDL.

II. RELATED WORK

Dynamism manifests in workflow environments in a variety of ways [4]. Our work focuses on data volatility, which van der Aalst describes as dynamism in the information perspective [3]. Dealing with changes of this type constitute only a small portion of the general adaptation problem. Indeed, the research literature on adaptation has been broad and encompasses many different types of volatility.

Earlier work focused on handling exceptions that occur in workflows [7]–[9]. These events often result in task failures. Tasks that fail return an exception interpreted by an exception handler. The handler resolves these failures by using either manual correction techniques or the more advanced event-condition-action (ECA) paradigm, where pre-constructed rules trigger a change in the workflow when exceptions takes place [10]. Typically, transaction constructs are employed such as task rollbacks and compensations [11], [12] to maintain correct and consistent workflows [5], [13]. However, changes to the task instance data was often overlooked in favor of guaranteeing that tasks interoperate.

Volatility in other aspects of the WSC has also received attention. Stohr and Zhao [14] focus on changes in the

organizational perspective. They devised a Business Process Adaptation Model, which decides how changes in business technologies may affect the needs of business process automation. Desai et al. [6] focus on adapting processes using handcrafted protocols (i.e. a set of rules that govern a business interaction) to alleviate problems of heterogeneity and to support autonomy among different WS providers. Van der Aalst et al. [15] addressed the problem of “dynamic change” – handling old instances in new workflow processes. The dynamic change problem is not relevant to this article, as we consider adapting WSCs that are executed on an individual, case by case basis.

Only recently, however, have researchers turned their efforts toward identification of change in the individual services’ parameters [16]. In Chafle et al. [17], [18] several alternate plans are pre-specified 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 are not used making the approach inefficient. Paques et al. [19] 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 for WSCs to adapt to changes in the data, it does not consider the costs of obtaining the revised data. Doshi et al. [20] adapt compositions using a technique that manages the dynamism of WSC environments through Bayesian learning. The process model parameters are updated based on previous interactions with the individual WSs and the composition is regenerated using these updates. This method is slow in updating the parameters, and the approach may result in plan re-computations that do not bring about any change in the WSC. Au et al. [21]–[23] introduce a framework that builds WSCs in the presence of data volatility. Using a reactive querying policy, they obtain current parameters of the WSC by querying WS providers 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 unnecessary computations.

Gotz and Mayer-Patel [24] incorporate multidimensional data adaptation using a metric similar to the value of information. They determine if new information may impact the utility of multimedia applications. Nanjangud et al. [25] and Charfi et al. [26] apply aspect-oriented programming to WSCs. Aspects are 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. [27] 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. [28] and Wu and Doshi [29] 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.

III. MOTIVATING SCENARIOS

In order to illustrate our approach we present two example scenarios involving WSCs.

A. MS XBOX Supply Chain

Our first scenario is a simplified version of the supply chain employed by Microsoft (MS) for the production of its Xbox 360 gaming console [30]. MS engages a variety of suppliers and contract manufacturers to deliver the components that are crucial to the production of the gaming console. Because MS outsources key manufacturing operations, it needs to retain tight control over those external processes to ensure that the suppliers and contract manufacturers meet service level agreements.

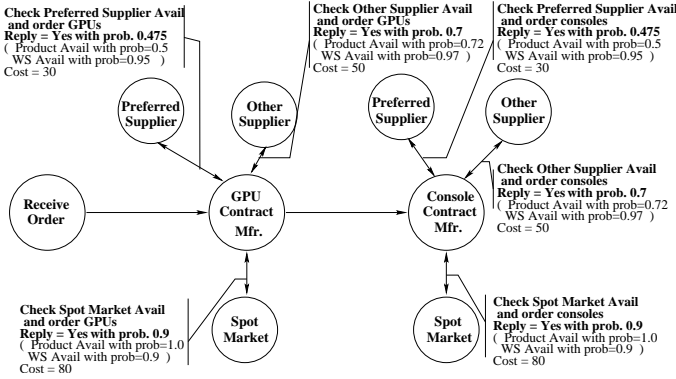


Fig. 1. Interactions between the business partners in the Microsoft Xbox 360 supply chain. Both the contracted manufacturers may select from a choice of suppliers. We have used example probability and cost values for the purpose of illustration.

In Fig. 1, we focus on a simplified supply chain in which MS chooses a contracted console manufacturer, who is responsible for assembling the console, and a contracted graphics processing unit (GPU) manufacturer who is responsible for building the advanced GPU chips. We assume that the invocations will be carried out in a sequential manner, beginning with the GPU followed by the console. Additionally, each of the manufacturers have the option to order their components from three different suppliers. They may order from a preferred supplier with which they usually interact. The manufacturers may also order the parts from other suppliers or resort to the spot market. A *costing analysis* reveals that the least cost will be incurred if the order is satisfied by the preferred supplier. The manufacturers will incur increasing costs as they try to fulfill the orders by procuring the console and GPU chips from another supplier and the spot market.

Clearly, MS and its manufacturers must choose from several candidate compositions. For example, they may initially attempt to satisfy the order of GPU chips from the preferred supplier. If the preferred supplier is unable to satisfy the order, the manufacturers may resort to ordering parts from another supplier. Another composition may involve bypassing the preferred supplier, since MS strongly believes that the preferred supplier will not satisfy the order. It may then initiate a status check on some other supplier. These example compositions reveal two factors for selecting the optimal one.

First, MS must accurately know the certainty with which the console and GPU chip orders will be satisfied by each of its supplier choices. Second, at each stage, rather than greedily selecting an action with the least cost, MS must select the action which is expected to be optimal over the long term.

B. Patient Transfer Clinical Pathway

A hospital receives a patient who has complained of a persistent fever. 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. For this example, 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 combination of high vacancy rate and least cost (see Fig. 2).

Similar to our previous example, several candidate 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 of the care givers, costs of treatment, and geographic proximity.

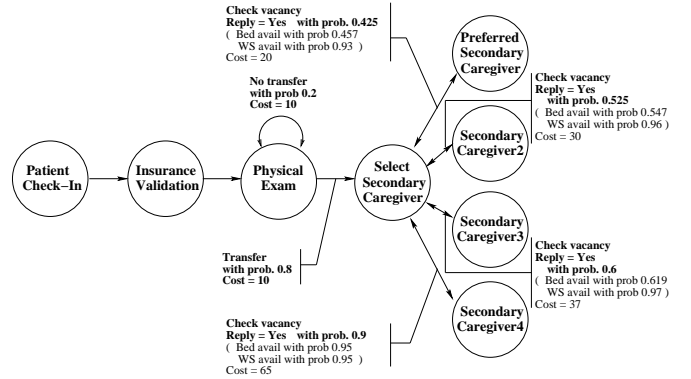


Fig. 2. The patient transfer clinical pathway for a primary caregiver. If a transfer is recommended, the WSC has a choice of selecting a secondary caregiver WS among many. As before, we have used example probability and cost values for illustration.

Both the supply chain and the patient transfer WSCs hinge, in part, on the rates of order satisfaction and vacancy rates, respectively. If the order satisfaction rate of the preferred supplier or the vacancy rate of the preferred secondary caregiver drops, the WSCs must adapt to remain cost-effective.

IV. BACKGROUND: WSC USING MDPs

For the purpose of illustration, we select a decision-theoretic planning technique for composing WSs [20]. However, our approach is applicable to any model based service composition technique such as [31] (for example, see [18]). Decision-theoretic planners such as MDPs model the WSC environment, WP , using a sextuplet:

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

where $S = \prod_{i=1}^k X^i$, S is the set of all possible states factored into a set, X , of k variables, $X = \{X^1, X^2, \dots, X^k\}$; 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 WSC.

In order to gain insight into the use of MDPs for WSC, let us model the supply chain of Section III-A as a MDP. The state of the composition is captured by the random variables – **GPU Preferred Supplier Availability**, **GPU Other Supplier Availability**, **GPU Spot Market Availability**, **Console Preferred Supplier Availability**, **Console Other Supplier Availability**, and **Console Spot Market Availability**. A state is then a conjunction of assignments of either *Yes*, *No*, or *Unknown* to each random variable. Actions are WS invocations, $A = \{\text{Check GPU Preferred Supplier Status, Check GPU Other Supplier Status, Check GPU Spot Market Status, Check Console Preferred Supplier Status, Check Console Other Supplier Status, Check Console Spot Market Status}\}$.

The transition function, T , models the non-deterministic effect of each action on some random variable(s). For example, invoking the WS Check GPU Preferred Supplier Status will cause **GPU Preferred Supplier Availability** to be assigned *Yes* with a probability of $T(\text{GPU Preferred Supplier Availability}=\text{Yes}|\text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability}=\text{Unknown})$. This rate of order satisfaction depends on two probabilities: (1) the probability that the GPU Preferred Supplier has sufficient GPUs for the order, and (2) the availability of the GPU Preferred Supplier’s WS interface. If the two availabilities are independent of one another³, we may view T as a product of these two probabilities: $T(\text{GPU Preferred Supplier Availability}=\text{Yes}|\text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability}=\text{Unknown}) = Pr(\text{GPU Preferred Supplier Product Availability}=\text{Yes}) \times \text{WS Availability}$. Similarly, the **GPU Preferred Supplier Availability** will be assigned *No* with a probability of $T(\text{GPU Preferred Supplier Availability}=\text{No}|\text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability}=\text{Unknown}) = Pr(\text{GPU Preferred Supplier Product Availability}=\text{No}) \times \text{WS Availability}$, and *Unknown* with a probability of $T(\text{GPU Preferred Supplier Availability}=\text{Unknown}|\text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability}=\text{Unknown}) = 1 - \text{WS Availability}$. 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 GPU Preferred Supplier WS and the cost of the GPU itself. We let H be some finite value which implies that the manufacturer 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 manufacturer has modeled its WSC problem as a MDP, it may apply standard MDP solution techniques to arrive at an optimal composition. These solution techniques revolve around the use of stochastic dynamic programming [32] for calculation of the optimal policy using *value iteration*:

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

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

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 composition, the optimal action for each state is the one which results in the minimum expected cost.

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

In Eq. 3, π_n^* is the optimal *policy* which is simply a mapping from states to actions for n steps to go, $\pi_n^* : S \rightarrow A$. The WSC is composed by performing the WS invocation prescribed by the policy given the state of the composition and observing the results of the actions. Doshi et al. [20] provide an algorithm for the composition. It takes the optimal policy, and the starting state of the WSC as input, and interleaves composition and execution. For each state encountered during the execution of the WSC, we refer to the policy of the MDP to recommend the current WS to invoke. The response of the service provides values for the random variables, effectively transitioning into a new state. This process is repeated until H steps are exhausted.

V. 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 preferred supplier’s services may increase 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 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. We present one such methodology next which earlier appeared in [1].

As we mentioned, we adopt a myopic approach to information revision, in which we query a single provider at a

³The two probabilities could be dependant on each other depending on the underlying business logic of the WS. For example, the GPU availability may influence the GPU Preferred Supplier’s decision to keep the WS active. For simplicity in our scenario, however, we assume that they are independent.

time for revised information. In the supply chain, this would translate to asking, say, only the preferred supplier for its current rates of order satisfaction (GPU and WS availability), as opposed to both the preferred supplier and the other supplier, simultaneously. The revised information may change the following transition probabilities, $T(\text{GPU Preferred Supplier Availability} = \text{Yes} \mid \text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability} = \text{Unknown})$, $T(\text{GPU Preferred Supplier Availability} = \text{No} \mid \text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability} = \text{Unknown})$, and $T(\text{GPU Preferred Supplier Availability} = \text{Unknown} \mid \text{Check GPU Preferred Supplier Status, GPU Preferred Supplier Availability} = \text{Unknown})$.

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 current belief distributions 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 current belief over the possible distributions. We illustrate below:

Example 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,

$$EV(s) = \int_{\langle p_1, p_2, 1-p \rangle} Pr(T'(\text{GPU Pref. Supp. Avail.} = \text{Yes/No/Unknown} \mid \text{Check GPU Pref. Supp. Status, Pref. Supp. Avail.} = \text{Unknown}) = \langle p_1, p_2, 1 - (p_1 + p_2) \rangle) V_{\pi^*}(s|T') d\mathbf{p}$$

assuming that the random variable **GPU Preferred Supplier Availability** assumes either *Yes*, *No*, or *Unknown* 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* (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) [33]. 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 WSC. 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:

Example In the supply chain example, the VPI provides a way to gauge the expected impact of knowing additional (previously unknown) parameters of WSs such as say, time to service failure and time to service repair, on the composition. In comparison, the VOC measures the expected impact of revised values of parameters that were previously considered while forming the initial composition, such as order satisfaction rate and service cost.

Analogous to VPI, the following proposition holds for VOC.

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

Proof: The proposition follows trivially if we find that $\forall s, \mathbf{p} \quad V_{\pi}(s|T') - V_{\pi^*}(s|T') \geq 0$, where $T'(\cdot|a, s') = \mathbf{p}$. By definition (Eq. 3), π^* is an optimal policy for the revised model. This implies that for any other policy, $\pi' \in \Pi \setminus \pi^*$, 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 $\Pi \setminus \pi^*$, or be equal to π^* .⁴ ■

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.

We may analogously apply the VOC methodology when the cost parameters of the services fluctuate. Instead of updating the transition function, T , we update the cost function, C . We obtain the cost distribution that may be queried, $C'(a, s)$, from the provider associated with WS a and use this distribution to find both $V_{\pi^*}(s|C')$ and $V_{\pi}(s|C')$. We may then use Eq. 5 analogously to find $VOC_{C'(a, s)}(s)$ and query if $VOC_{C'(a, s)}(s) > QueryCost(C'(a, s))$.

VI. MITIGATING COMPUTATIONAL COMPLEXITY OF VOC

Calculating the VOC as shown in Eq. 5 is computationally intensive. The probability \mathbf{p} represents a revised probability of transition on performing a particular action. To calculate the VOC, we must compute the revised values, $V_{\pi}(s|T')$, and $V_{\pi^*}(s|T')$ for all possible \mathbf{p} and average over their difference based on our distribution over \mathbf{p} . We note that computing

⁴Note that we assume that the revised information is received *without* noise.

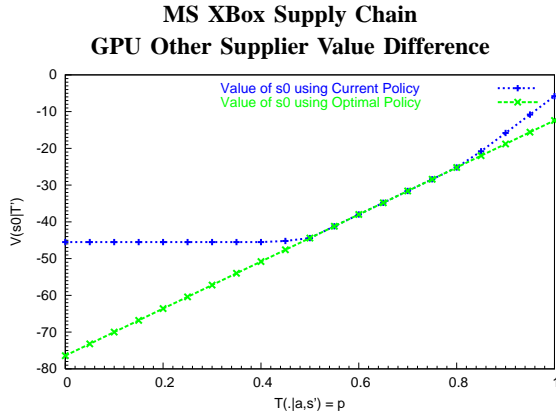


Fig. 3. The plot of $V_{\pi^*}(s|T')$ and $V_{\pi}(s|T')$ for the GPU other supplier in the start state of the MS Xbox supply chain.

$V_{\pi}(s|T')$, which represents the expected cost of following the policy π from state s is straightforward since it does not involve the minimization operation. However, the revised value function $V_{\pi^*}(s|T')$ is computed using the standard value iteration as defined in Eq. 1.

One way to mitigate the computational cost of calculating VOC is to reduce the range of \mathbf{p} over which the averaging is carried out. We describe a way to achieve this next.

A. Pruned Averaging

In order to reduce the effective range of \mathbf{p} , and more generally the effective range of possible values of parameters, we find out those values of \mathbf{p} for which the expected value of the optimal policy given the revised probability, $V_{\pi^*}(s|T')$, remains unchanged from that of the existing policy. This typically occurs because the revised parameters did not alter the existing optimal policy. These values of \mathbf{p} do not contribute to the VOC and therefore may be ignored without affecting the VOC. Obviously, there is at least one point in the range of \mathbf{p} which does not contribute to the VOC and may be pruned. Proposition 2 formalizes this observation:

Proposition 2: Given state s of the composition guided by policy π , and WS a , there exists at least one $T'(\cdot|a, s') = \mathbf{p}'$, for which $V_{\pi^*}(s|T') = V_{\pi}(s|T')$.

Proof: Let \mathbf{p}' be the current value of the parameter of the composition under consideration. Given that all other parameters remain unchanged, the optimal policy, π^* , when $T'(\cdot|a, s') = \mathbf{p}'$ is also the current policy. Therefore, $V_{\pi^*}(s|T') = V_{\pi}(s|T')$. ■

Although Proposition 2 points to the existence of at least one \mathbf{p} in the integral range of Eq. 5 that does not contribute to the VOC, typically this range tends to be larger. For example, in Fig. 3, we show the plots of $V_{\pi^*}(s|T')$ and $V_{\pi}(s|T')$ as \mathbf{p} is varied, for the supply chain problem where the WS under consideration is Check GPU Other Supplier Status in its initial state (s_0). Notice a significantly large region of overlap between the two plots – the range of \mathbf{p} spanning the overlap does not contribute to the VOC and may be ignored. As an aside, we observe that $V_{\pi^*}(s|T')$ is monotonically non-decreasing and later diverges from $V_{\pi}(s|T')$. This is because the rate of order satisfaction of the other supplier has improved

Error Function for GPU Other Supplier

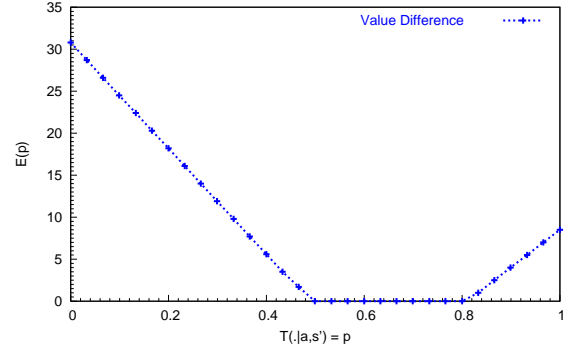


Fig. 4. The error landscape where the error is $V_{\pi^*}(s|T') - V_{\pi}(s|T')$ for the GPU other supplier. The plateau signifies the region of zero error.

to an extent where the new supplier replaces the preferred supplier as the option of choice in the optimal policy.

All that remains now is to find the boundary points, say \mathbf{p}_{\min} and \mathbf{p}_{\max} , of the region of overlap and prune the range $[\mathbf{p}_{\min}, \mathbf{p}_{\max}]$ from consideration while calculating the VOC. Formally,

$$\begin{aligned} & \int_{\mathbf{p}} Pr(T'(\cdot|a, s') = \mathbf{p}) \overbrace{[V_{\pi}(s|T') - V_{\pi^*}(s|T')]}^{\Delta V} d\mathbf{p} \\ &= \int_0^{\mathbf{p}_{\min}} Pr(T'(\cdot|a, s') = \mathbf{p}) \Delta V d\mathbf{p} + \int_{\mathbf{p}_{\max}}^1 Pr(T'(\cdot|a, s') = \mathbf{p}) \Delta V d\mathbf{p} + \\ & \quad \int_{\mathbf{p}_{\min}}^{\mathbf{p}_{\max}} Pr(T'(\cdot|a, s') = \mathbf{p}) \Delta V d\mathbf{p} \\ &= \int_0^{\mathbf{p}_{\min}} Pr(T'(\cdot|a, s') = \mathbf{p}) \Delta V d\mathbf{p} + \int_{\mathbf{p}_{\max}}^1 Pr(T'(\cdot|a, s') = \mathbf{p}) \Delta V d\mathbf{p} \\ & \quad (V_{\pi}(s|T') - V_{\pi^*}(s|T') \text{ is } 0 \text{ for } \mathbf{p}_{\min} \text{ to } \mathbf{p}_{\max}) \end{aligned}$$

In the next subsection, we present a fast numerical method to compute the boundary points.

B. Computing the Intersection Points Using Gradient Descent

The intersection points, \mathbf{p}_{\min} and \mathbf{p}_{\max} , may be computed in different ways. We could formulate a quadratic program (QP) that would minimize the difference between $V_{\pi^*}(s|T')$ and $V_{\pi}(s|T')$ and return the maximum and minimum values of \mathbf{p} , where the difference is the lowest ($= 0$). However, not only is the formulation of such a QP complex, but solution methods for general purpose QPs are not yet well-developed.

Alternately, we may view the problem of finding the intersection points as that of descending down the error surface until we reach a plateau. The *error* is the difference between $V_{\pi^*}(s|T')$ and $V_{\pi}(s|T')$. We show the error landscapes for the plots of Fig. 3 in Fig. 4. The typical drawback of this approach, called gradient descent [34] – getting stranded on local rather than global minimas – is not a concern here because of the presence of always a single minimum in the landscape.

Within the gradient descent approach, we update the parameter, \mathbf{p} , in the following way:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p} \quad ; \quad \Delta \mathbf{p} = -\eta \frac{\partial E(\mathbf{p})}{\partial \mathbf{p}} \quad (6)$$

Here, η is the step size, $0 \leq \eta \leq 1$, the negative sign indicates that we take a step in the direction of the reducing gradient, and the error function, $E(\mathbf{p}) = V_{\pi^*}(s|T') - V_{\pi}(s|T')$,

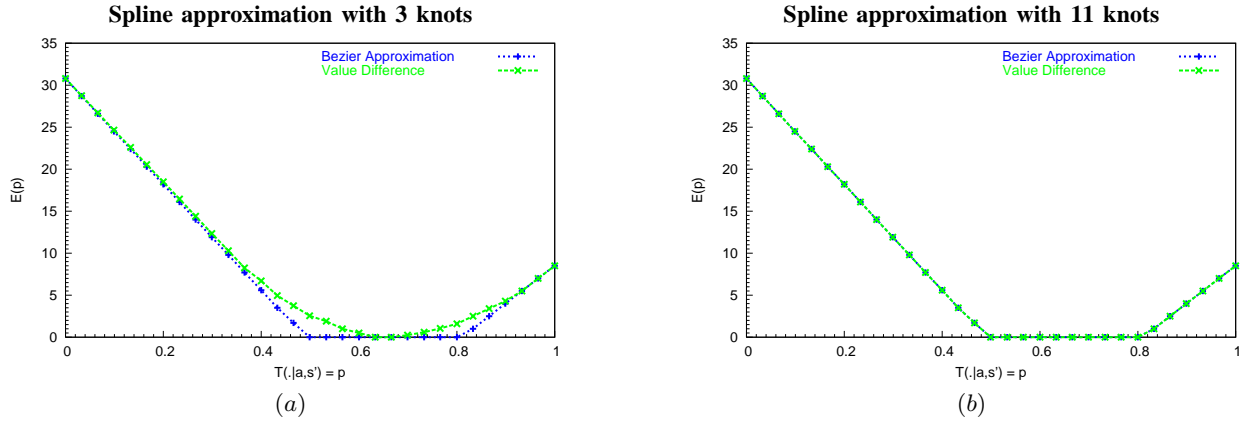


Fig. 5. The error function for the GPU Other Supplier approximated using splines of (a) 3 knots (2 Bezier curves) and (b) 11 knots (10 Bezier curves).

$T'(\cdot|a, s') = \mathbf{p}$. Beginning from an initial value, we continuously update \mathbf{p} until the difference between the revised and the previous values of \mathbf{p} becomes very small.

Computing the partial derivative of the error function, $\frac{\partial E(\mathbf{p})}{\partial \mathbf{p}}$, is difficult because of the recursive nature of the Bellman equation that defines the value function (Eq. 1).

To avoid this arduous step, we propose to approximate the error surface with a set of basis functions that are relatively simple to differentiate. The particular shape of the error surface and the need for functions that are easily differentiable, suggests that *polynomial splines* [35] may serve as good candidates. A spline is a general piecewise function where the pieces are polynomials and is capable of approximating any shape up to arbitrary accuracy. Because the pieces are polynomials, a spline is relatively simple to differentiate.

Formally, a polynomial spline is defined as, $S : [a, b] \rightarrow \mathbb{R}$, which consists of polynomial pieces, $P_i : [t_i, t_{i+1}) \rightarrow \mathbb{R}$, where: $a = t_0 < t_1 < \dots < t_{k-2} < t_{k-1} = b$. The k points, $t_j : j = 0 \dots k-1$ are referred to as the *knot vector*. Knots represent the border points of pieces of the polynomial.

In our application, we uniformly select k equidistant knots. Selecting the j^{th} knot involves computing, $V_{\pi^*}(s|T' = \mathbf{p}_j) - V_{\pi^*}(s|T' = \mathbf{p}_j)$, where \mathbf{p}_j is the x-coordinate of the knot.

Given the knots for our spline, we may now formulate the polynomials at each of the intervals. A popular parametric representation for the polynomials is the *Bèzier* curve of degree d , which may be generalized as follows: Given points, P_0, P_1, \dots, P_d , the Bèzier curve is:

$$B(\theta) = \sum_{i=0}^d \binom{d}{i} P_i (1-\theta)^{d-i} \theta^i \quad (7)$$

where P_0 and P_d are the evaluations of consecutive knots, t_j and t_{j+1} , respectively, and θ is the parameter, $0 \leq \theta \leq 1$. For a Bèzier curve of degree d , $d+1$ points are needed. Of course, the more knots selected and the higher the degree of the Bèzier curve, the more accurate the approximation will be.

In Fig. 5, we show the original error surface of Fig. 4 as well as the approximation using a polynomial spline where the polynomial pieces are Bèzier curves of degree at most two. We selected 3 equidistant knots (5 points in all) for computing the spline shown in Fig. 5(a) and 11 knots (21 points) for computing the spline of Fig. 5(b). The latter spline closely

approximates the original error surface, though at the expense of computing a larger knot vector.

Differentiating the polynomial spline piecewise and utilizing Eq. 6, we get the following piecewise rule for updating the parameter \mathbf{p} :

$$p \leftarrow p - \eta \begin{cases} 2 \frac{(P_{0,y}^1 - 2P_{1,y}^1 + P_{2,y}^1)}{(4(P_{1,x}^1)^2 - 8P_{1,x}^1 P_{0,x}^1 + 4(P_{0,x}^1)^2)} p - & t_0 \leq p < t_1 \\ \frac{2P_{0,x}^1 (P_{0,y}^1 - 2P_{1,y}^1 + P_{2,y}^1)}{(4(P_{1,x}^1)^2 - 8P_{1,x}^1 P_{0,x}^1 + 4(P_{0,x}^1)^2)} - & \\ \frac{(P_{1,y}^1 - P_{0,y}^1)}{(P_{1,x}^1 - P_{0,x}^1)} & \\ \dots & \dots \\ 2 \frac{(P_{0,y}^{k-1} - 2P_{1,y}^{k-1} + P_{2,y}^{k-1})}{(4(P_{1,x}^{k-1})^2 - 8P_{1,x}^{k-1} P_{0,x}^{k-1} + 4(P_{0,x}^{k-1})^2)} p - & t_{k-1} \leq p < t_k \\ \frac{2P_{0,x}^{k-1} (P_{0,y}^{k-1} - 2P_{1,y}^{k-1} + P_{2,y}^{k-1})}{(4(P_{1,x}^{k-1})^2 - 8P_{1,x}^{k-1} P_{0,x}^{k-1} + 4(P_{0,x}^{k-1})^2)} - & \\ \frac{(P_{1,y}^{k-1} - P_{0,y}^{k-1})}{(P_{1,x}^{k-1} - P_{0,x}^{k-1})} & \end{cases}$$

The derivation of this update rule is given in Appendix 1. Depending on the current value of \mathbf{p} , the appropriate piece from the piecewise function will be selected.

Beginning from an initial value, the gradient descent rule shown previously will update \mathbf{p} until the difference between the revised and the previous values of \mathbf{p} becomes very small indicating that the boundary of the plateau has been reached. We observe that a large step size, η , may cause the gradient descent to reach the plateau faster, but it may also result in overstepping the boundary region. Consequently, larger η may result in undesirable oscillations.

VII. WEB SERVICE COMPOSITION WITH VOC

We need only to make a small change to the algorithm outlined in Section IV 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.

A. Algorithm

We show the algorithm for an adaptive WSC in Fig. 6.

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

$$a^* = \underset{a \in A}{\operatorname{argmax}} \operatorname{VOC}_{T'(\cdot|a, s')}(s) \quad (8)$$

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

$$\operatorname{VOC}^*(s) = \underset{a \in A}{\operatorname{max}} \operatorname{VOC}_{T'(\cdot|a, s')}(s) \quad (9)$$

If $\operatorname{VOC}^*(s)$ exceeds the query cost $\operatorname{QueryCost}(T'(\cdot|a, s'))$, the query is issued. The WSC receives the new parameter values and uses them to recompute π^* for all states S .

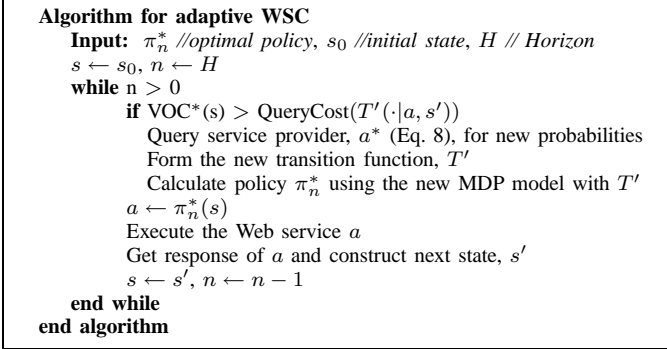


Fig. 6. Algorithm for adapting a WSC to revised information.

Recomputing the policy is expensive – if N is the number of possible values a random variable X^k can take, then its complexity is $\mathcal{O}(N^{2|X||A||H|})$. However, recomputing the entire policy is necessary in general, as any state may be subsequently reached in the WSC, including previously visited states. We are currently investigating special cases where partial recomputation may be sufficient.

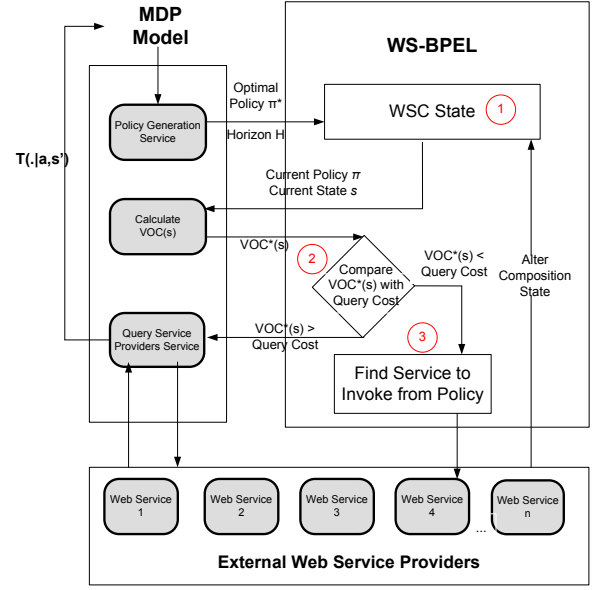
B. Experiments

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 four heuristic approaches.

1) *Architecture:* The algorithm described in Fig. 6 is implemented as a WS-BPEL flow and all WSs were implemented using WSDL.

To the WS-BPEL flow, we give the optimal policy, 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. 7(a).

Implementation of MS Xbox Supply Chain



(a)



(b)

Fig. 7. SOA for implementing our adaptive WSC. (a) demonstrates the interaction of the composition with our pre-constructed internal services. (b) shows a sample of the BPEL markup for the MS Xbox supply chain scenario. Labels (1) (2) and (3) in (a) correspond to its associated markup in (b).

Within our SOA, we provide internal WSs for generating the policy from the MDP model, and for computing the VOC with pruned averaging. If the VOC exceeds the cost of querying a provider (this cost is also provided as an input), the WS-BPEL flow invokes a special WS whose function is to query the provider's information-providing WSs for revised information.

This information is used to form and solve a new MDP. The new policy is fed back to the WS-BPEL flow. The policy is used by the WSC to invoke the prescribed external WS. The WS response is used to formulate the next state of the WSC. This procedure is repeated until H steps have been 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. 7(b). First, note that the following constructs must be added to implement the VOC algorithm:

- data structures for state and policy,
- tasks invoking a VOC computation service, comparing the VOC to the query cost, and regenerating the policy
- a task that will invoke the external services providers as recommended by the policy.

As outlined by section (1) in Fig. 7(b), 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 $\langle \textit{while} \rangle$ condition corresponds to the while loop in Fig. 6. Each state has an associated $\langle \textit{switch} \rangle \langle \textit{case} \rangle$ construct. In each $\langle \textit{case} \rangle$, the WSC invokes the 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 queried parameters are integrated into the MDP, which invokes the policy generator service, returning the new optimal policy thereby replacing the old policy of the WSC. The new policy is then used to recommend the optimal service to invoke in the state. This process repeats until the composition has terminated.

2) *Performance Evaluation*: We utilized the MS Xbox supply chain and the clinical patient transfer scenarios (Section III) for our evaluations. For the supply chain example, we simulated querying the suppliers for their current percentage of order satisfaction (GPU and WS availability) while in the patient transfer pathway, we queried the secondary caregivers for their current vacancy rates.⁵

We model the MS-XBOX 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. 8(a) shows the beta densities that represent the manufacturer’s distribution over the rate of order satisfaction by, say the GPU Preferred Supplier, ie. $Pr(\text{GPU Pref. Supp. Avail.}=\text{Yes} \mid \text{Check GPU Pref. Supp.}, \text{GPU Pref. Supp. Avail.}=\text{Unknown})$, and analogously for the other suppliers.

We emphasize that these densities are projections of the more complex ones that would account for all the factors that may influence a supplier’s ability to satisfy an order, such as the time that an order is placed and quantity of the order. Means of the densities reveal that the preferred supplier tends to be less reliable in satisfying orders than other suppliers.

Fig. 8(b) shows the density plots over the probability of a vacancy with the preferred and other secondary caregivers.

In Fig. 9 we compare the VOC-driven selective querying based adaptation with four other strategies with respect to the average cost incurred from executing the adapted WSCs, as the cost of querying the service providers is increased. Our methodology consisted of running a trial of 50 independent instances of each composition within a simulated volatile environment, where the queried parameters of the service providers were distributed according to the density plots in Fig. 8. The data points shown in Fig. 9 are averaged over 5 such trials. We ensured that the compositions using each of the five strategies received similar responses from the service providers.

The four approaches utilized for adaptation are:

- 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 each time for querying for revised information.
- 3) **Intermittent querying** We begin by querying services every alternate instance. As the costs of querying increase, we reduce the frequency with which we query.
- 4) **Largest difference** This approach utilizes the distributions of the services parameters shown in Fig. 8. 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. In Figs. 9(a) and (b), we show the results for the supply chain and patient transfer 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, 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 (in particular, see Fig. 9(b)), 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 *physical exam* service in the patient transfer WSC, does not affect the composition though the approach will query it and incur the query cost. In addition, the difference in parameter values is not comparable to query costs. In summary, a WSC that is adapted using VOC performs better (incurs less average cost)

⁵Of course, the rate of order satisfaction in the supply chain, for example, would depend on the quantity of the order and other factors; we assume that these will be provided to the suppliers.

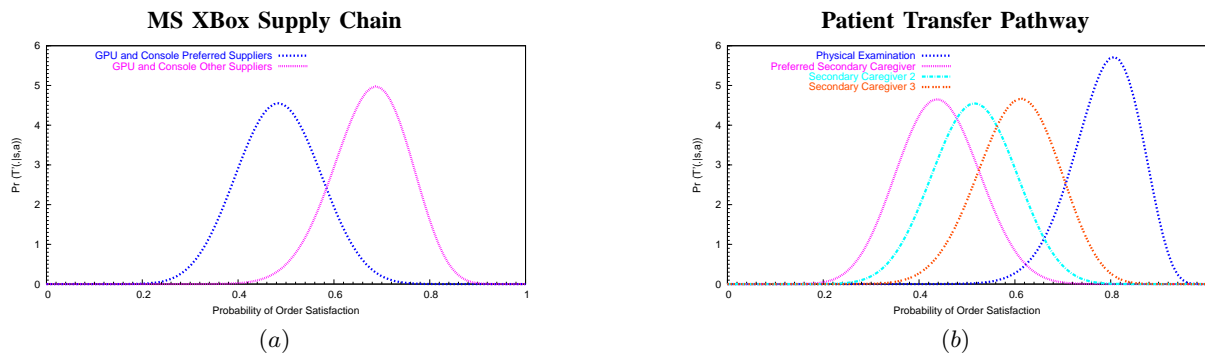


Fig. 8. The probability density functions representing (a) MS’s belief over the GPU and console suppliers’ rates of order satisfaction in the Xbox supply chain scenario; (note that both the GPU and Console Suppliers have the same curves because their parameters for our experiments were identical) (b) the primary caregiver’s beliefs over the secondary caregivers’ probabilities of having a vacancy.

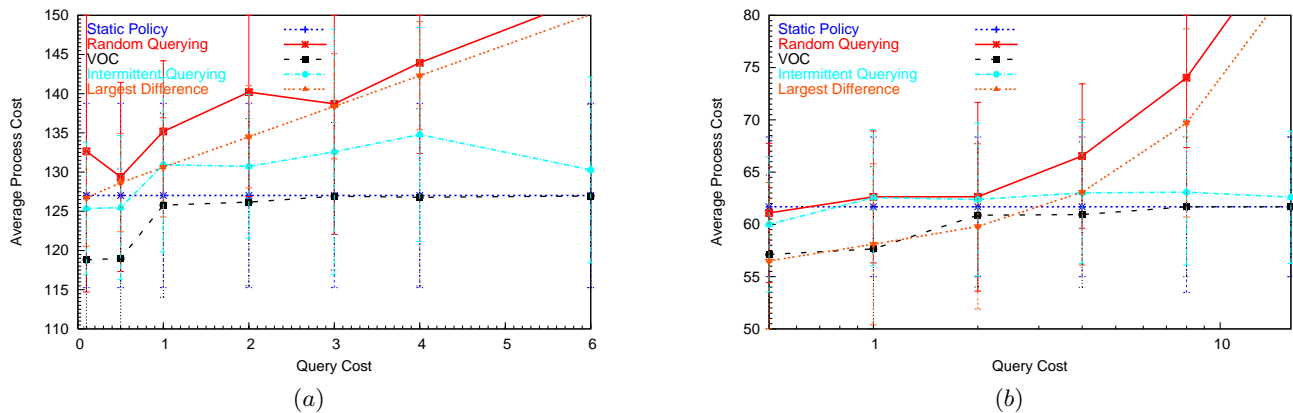


Fig. 9. Comparisons of the VOC based adaptive WSC with the static policy and other querying approaches for (a) MS Xbox supply chain, and (b) patient transfer scenarios. Lower average process cost indicates better performance. The deviation bars demonstrate the variance caused by a randomized environment.

because only significant changes to the WSC are carried out while simultaneously avoiding frequent costly queries.

3) *Computational Speedup*: In Fig. 10, we give the average run time of executing and adapting the WSC, as described in Fig. 6. Note that these times include the computation of VOC (Eq. 5). We first observe that an adaptive WSC that uses VOC takes an order of magnitude more time than one that randomly selects service providers to query for revised information. However, as we mentioned previously, the adapted composition in the latter case is significantly more costly in comparison to the VOC based approach.

Problem	Query Random	VOC	VOC with Pruned Averaging
Supply Chain	17s	285s	115s
Patient Transfer	15s	101s	34s

Fig. 10. A comparison of the average execution times of the MS Xbox supply chain and patient transfer WSCs. By using pruned averaging, we were able to cut down the run time of adapting a composition by approximately a factor of 3.

Furthermore, by pruning the effective parameter values as described in Section VI, we reduced the execution times of the VOC based adaptation by approximately a factor of three. Note that these run times include the time consumed in performing the gradient descent. The speedup was consistent across both the example scenarios. We present a way to further reduce the complexity of VOC based adaptation in the next section.

VIII. OPTIMIZING VOC* USING EXPIRATION TIMES

As we mentioned previously, in order to select a service provider for querying for revised information, computing VOC^* required iterating over all the WSs. For large WSCs, there could be several participating WSs, making the process of selection computationally intensive. To address this challenge, we use the insight that service providers are often able to guarantee certain characteristics during execution. Guarantees can be described in a service level agreement (SLA). SLAs are contracts between providers and users of the providers’ services on the level of service that is expected during its term. Here, we exploit a particular guarantee, parameter expiration times, to further reduce VOC’s computational overhead.

A. VOC^E : VOC with Expiration Times

We use the insight that service providers are often able to guarantee that their order satisfaction rates and other parameters will remain fixed for some time, t_{exp} , after which they may vary. WS providers may define t_{exp} in a WS-Agreement document as we show later ⁶.

Given a way to keep track of which WSs’ parameters have expired, we may compute the VOC for only those services whose guarantees have expired and select among them [2].

⁶WS-Agreement: www.ogf.org/documents/GFD.107.pdf

then we query the provider for the new parameters, which form the new T' in the WSC model. We also add the time taken to perform the $VOC^\mathcal{E}$ calculations to the cumulative time counter associated with each WS (lines 12-16). On querying, in addition to obtaining the possibly revised information, we also obtain the new expiration times for the information. Thus, the counter for the WS that is queried is reset.

We observe that querying for information is not a constant time step operation, but must take into account the time taken for the request to reach the provider, the provider's information-providing WS to complete its computations, and for the response to arrive back at the composition. We denote the total time consumed in querying as t_{QLag} , which is depicted in Fig. 13⁷.

The revised information is integrated into the WSC model and a new policy is recomputed to maintain optimality of the WSC. However, re-computation of the policy is not always necessary, and runtime changes could be made to the WSC. Here, the time counters must be updated again to account for the time taken to recalculate the policy. Finally, the queried WS is removed from \mathcal{E} (lines 22-27). We observe that the times, t_{QLag} and $t_{\pi_n^*}$ could be calculated in *real-time* (online) using timestamps before and after the calculations.

Of course, if the query cost exceeds $VOC^\mathcal{E}$, then we ignore the previously mentioned steps, and simply invoke the WS that the original policy recommends. Obtaining a response from the invoked WS may not be a constant time operation but may depend on external factors, as shown in Fig. 13. Let $t_{Response}$ be the time elapsed, then this time is added to all the cumulative time counters (lines 29-34).

B. Theoretical Results

We first show that given an identical input, adaptation using $VOC^\mathcal{E}$ results in the same WSC as compared to adaptation using VOC^* (Eq. 9).

Proposition 3 (Correctness): Given identical policies and start states, adaptation using $VOC^\mathcal{E}$ and VOC^* generate identical WSCs.

Proof: We begin with the definition of VOC^* (Eq. 9):

$$\begin{aligned} VOC^*(s) &= \max_{a \in A} VOC_{T'(\cdot|a,s')}(s) \\ &= \max_{a \in \mathcal{E} \cup \bar{\mathcal{E}}} VOC_{T'(\cdot|a,s')}(s) \end{aligned}$$

where $\bar{\mathcal{E}}$ is the complement of \mathcal{E} as mentioned previously.

We consider two cases: (i) If the WS with the maximum VOC, selected for querying, has expired parameters, $a^* \in \mathcal{E}$, then $VOC^* = VOC^\mathcal{E}$ for every state, and the WSC will be adapted identically to when $VOC^\mathcal{E}$ is used. On the other hand, (ii) if the WS parameters associated with the maximum VOC has not expired, then since the revised information is guaranteed to be unchanged, the belief distribution over parameters collapses to a point estimate, $VOC^* = 0$, and the WSC remains unchanged. Thus, for both the cases, the resulting WSC will be identical to the one generated when $VOC^\mathcal{E}$ is used. ■

⁷For simplicity, we assume that $t_{QLag} < t_{exp}$, so as to avoid having revised values expire before they are received. If this condition were not true, it would be best not to query.

We derive the complexity of the improvement next.

Proposition 4 (Improvement of Time Complexity by $VOC^\mathcal{E}$): Let N denote the number of possible values a random variable X can take. The worst-case complexity of adaptation using $VOC^\mathcal{E}$, as performed by the algorithm shown in Fig. 11, is:

$$\mathcal{O}(H(N^{2|X}|A|^2|H| + t_{QLag} + t_{Response})).$$

Here, the complexity is quadratic in the number of services ($|A|$). The best-case complexity of adaptation using $VOC^\mathcal{E}$ is:

$$\Omega(H(N^{2|X}|A||H| + t_{Response})),$$

which is linear in $|A|$. However, the tight-bound complexity of adaptation using traditional VOC is:

$$\Theta(H(N^{2|X}|A|^2|H| + t_{QLag} + t_{Response})).$$

Note that here, the worst case complexity is the same as the best case complexity, both of which are quadratic in $|A|$.

Proof: We refer to the algorithm for adaptation using $VOC^\mathcal{E}$ shown in Fig. 11. The outer while loop (line 4) will terminate when the composition has completed (taking at most H steps).

Within the body of the loop we focus on three operations in particular. First, lines 6-11 update the set of expired services, \mathcal{E} . Here, a loop iterates over all WSs (ie, $|A|$) effectively having an execution time in the order of $\mathcal{O}(|A|)$. However, each pass of the loop calls the *AddExpiredServices* procedure (Fig 12), which terminates when no more services are added to \mathcal{E} . In the worst case, this procedure will add one service to \mathcal{E} , and then $t_{VOC^\mathcal{E}}$ will increase such that one more service will expire, in which case another service will be added to \mathcal{E} ; this process is repeated until all the services are added. We may then write the following recurrence for the runtime of this procedure:

$$T(|A|) = \mathcal{O}(|A|) + T(|A| - 1) \quad (11)$$

Eq. 11 shows that each pass of the loop takes $\mathcal{O}(|A|)$ and, in the worst case, one service will be added to \mathcal{E} for each pass. This recurrence will run in $\mathcal{O}(|A|^2)$ time. Subsequently, lines 6-11 will take $\mathcal{O}(|A|^3)$. Second, line 12 involves a calculation of $VOC^\mathcal{E}$, which in the worst case collapses to a VOC^* calculation (when all services have expired). VOC^* , as mentioned in Eq. 9, is the maximum VOC over all services involved in the WSC, so $|A|$ VOC calculations are required. Each VOC calculation involves a comparison between values produced by the optimal policy and the current policy in the changed environment. With $|X|$ variables having maximum values N , the maximum state space size is $N^{|X|}$. Thus, solving the MDP and finding the optimal policy of a WSC takes $\mathcal{O}(N^{2|X}|A|H)$ time. Similarly, recalculation of the policy in line 23 will also take this time. In total, $VOC^\mathcal{E}$ will run in $\mathcal{O}(N^{2|X}|A|^2H)$ time. Finally, we must also consider t_{QLag} (line 20) and $t_{response}$ (line 33), as these are external to the WSC and independent of the VOC calculations.

Thus the total runtime complexity is:

$$\begin{aligned} \mathcal{O}(VOC^\mathcal{E}) &= \mathcal{O}(|N|^{2|X}|A|^2H^2) + \mathcal{O}(H \times |A|^3) + \\ &\quad \mathcal{O}(H \times t_{QLag}) + \mathcal{O}(H \times t_{Response}) \end{aligned}$$

We can eliminate the second term term because $N^{2|X|} \gg |A|$. So now we are left with the following complexity:

$$\mathcal{O}(VOC^{\mathcal{E}}) = \mathcal{O}(N^{2|X|}|A|^2H^2) + \mathcal{O}(H \times t_{QLag}) + \mathcal{O}(H \times t_{Response})$$

which may be rewritten as:

$$\mathcal{O}(H(N^{2|X|}|A|^2H + t_{QLag} + t_{Response}))$$

However, lines 12-28 are not necessarily executed for each state in the WSC. It is possible that a service has not expired, thus $|\mathcal{E}|$ is not equal to $|A|$. In these situations, the intensive computations required at lines 22 and 23 can be ignored. In the best case, none of these computations are required, yielding:

$$\Omega(VOC^{\mathcal{E}}) = \Omega(H(N^{2|X|}|A|H + t_{Response}))$$

Note that since none of the services will be queried, the cost of t_{QLag} may also be removed.

Traditional VOC has a worst case scenario equivalent to the worst case of $VOC^{\mathcal{E}}$:

$$\mathcal{O}(VOC) = \mathcal{O}(H(N^{2|X|}|A|^2H + t_{QLag} + t_{Response}))$$

However, using traditional VOC requires all WSs be polled at *every* time step of the WSC. That is, lines 12-28 will be executed in every step and $|\mathcal{E}|$ will be equal to $|A|$. Also, the query lag time t_{QLag} must be added to the complexity. Thus, the best case scenario will resemble the worst case.

$$\Omega(VOC) = \Omega(H(N^{2|X|}|A|^2H + t_{QLag} + t_{Response}))$$

Because the best case and the worst case complexities are the same, we may give the tight-bound complexity of:

$$\Theta(VOC) = \Theta(H(N^{2|X|}|A|^2H + t_{QLag} + t_{Response}))$$

IX. EXPERIMENTS

The algorithm described in Fig. 11 uses the architecture depicted in Fig. 7(a). Within our SOA, we provide internal WSs for solving the MDP model of the WSC problem and generating the policy, and computing the $VOC^{\mathcal{E}}$. If the $VOC^{\mathcal{E}}(s)$ 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 information-providing WSs for revised information and the new expiration times. 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 WSC. This procedure continues until the steps are exhausted.

The objective of our experimental evaluation is to show that the average execution time of the WSC adapted using $VOC^{\mathcal{E}}$ is less than when the WSC is adapted using VOC^* , and varies intuitively as the expiration times vary.

We again utilized the MS Xbox supply chain and the clinical patient transfer scenarios (Section III) for our evaluations. For the supply chain example, we queried the suppliers for their current percentage of order satisfaction while in the

```

<wsag:Agreement Name="xs:MS Xbox GPU Contract">
...
<wsag:Context>
  <wsag:ServiceProvider>
    xs:GPUPreferredSupplierURI
  </wsag:ServiceProvider>
  <wsag:ExpirationTime>11:59 27 Jan 2009</wsag:ExpirationTime>
  <wsag:TemplateId>...</wsag:TemplateId>
  <wsag:TemplateName>...</wsag:TemplateName>
</wsag:Context>
<wsag:Terms>
  <wsag:All>
    <wsag:ServiceDescriptionTerm
      wsag:Name="Rate of GPU Satisfaction"
      wsag:ServiceName="Order GPUs" >
      <job:SatisfactionRate>0.4</job:SatisfactionRate>
    </wsag:ServiceDescriptionTerm>
  ...
</wsag:All>
</wsag:Terms>
</wsag:Agreement>

```

Fig. 14. A WS-Agreement document showing the agreed upon expiration time and the rate of order satisfaction of a supplier for the Xbox supply chain.

patient transfer pathway, we queried the secondary caregivers for their current vacancy rates. In addition to the revised information about their services, the providers also guarantee a duration over which the WS parameter values will remain fixed.⁸ These distributions may be provided by the service providers using service-level agreements drawn up using, say, the WS-Agreement specification.

Figure 14 shows a part of an agreement between MS and the preferred GPU supplier in the Xbox scenario. t_{exp} is defined within the $\langle ExpirationTime \rangle$ within the $\langle Context \rangle$ tag. Here, the agreement will expire on January 27, 2009. Further in the document, the $\langle ServiceDescriptionTerm \rangle$ within the $\langle Terms \rangle$ tag defines the provider's rate of GPU order satisfaction (probability of 0.4). Thus, MS and the contracted GPU manager have agreed that any order of GPUs from MS will be satisfied 40 percent of the time until the agreement is voided on January 27, 2009.

MS Xbox Services	Cost	t_{QLag} (s)	$t_{Response}$ (s)
GPUPreferredSupplier	30	1	5
GPUOtherSupplier	50	1	4
GPUSpotMarket	80	1	3
ConsolePreferredSupplier	30	1	5
ConsoleOtherSupplier	50	1	4
ConsoleSpotMarket	80	1	3
Patient Transfer Services	Cost	t_{QLag} (s)	$t_{Response}$ (s)
InsuranceValidation	10	1	1
PhysicalExam	10	1	2
PreferredSecondaryCaregiver	20	1	3
SecondaryCaregiver2	30	1	3
SecondaryCaregiver3	37	1	3
SecondaryCaregiver4	65	1	3

Fig. 15. Costs, t_{QLag} , and $t_{Response}$ for the WSs in the MS Xbox supply chain and Patient Transfer scenarios.

For those services when their information expires, we model the manufacturer's and primary caregiver's beliefs over their possible parameter values, $(Pr(T'(\cdot|a, s') = \mathbf{p}))$ in Eq. 5) using the beta functions outlined previously in Fig. 8. For those services whose revised information has not expired, the

⁸In the real world, an example response by a supplier to a query could be, "We will guarantee meeting 2 of every 3 orders submitted through our WS interface for the next six months".

manufacturer’s and caregiver’s beliefs could be seen as Dirac-delta functions, with the non-zero probability value fixed at the probability, \mathbf{p} , that was provided at the time of query. Thus, for this case, the $VOC(s) = 0$ at any state of the WSC.

In order to perform the evaluations, we *simulated* a volatile business environment for each of the two problem domains. As before, for the supply chain, the rates of order satisfaction for the preferred and other suppliers were assumed to vary according to the density plots in Fig. 8. The expiration times were upper bound to a large time interval and randomly selected within the bound. The rates of order satisfaction remained fixed until the corresponding expiration times elapsed, after which, on query, new expiration times were randomly selected. Other parameters of the environment such as the WS invocation costs, t_{QLag} , and $t_{Response}$, are as given in the Figure 15. The environment for the patient transfer problem was simulated analogously (see Fig. 15).

In Figs. 16(a) and (b), we compare the run times for generating and executing the WSC. We compare the execution time of a WSC without any adaptation, with the execution time of a composition adapted using VOC^* (with pruning), and the execution time of a composition adapted using VOC^E (Fig. 11) (also with pruning). As we increase the expiration times associated with the revised information obtained from the providers, the composition execution time when adapted using VOC^E *decreases*. Notice that it is upper bounded by the execution times of a composition adapted using VOC^* , and lower bounded by the run times of a composition without adaptation. This is intuitive because VOC^* involves considering *all* participating WSs for querying, while no such computations are carried out in a composition without adaptation. Both these execution times are invariant with respect to the expiration times. Our results demonstrate the inverse relationship between expiration times and computational effort expended on adaptation. We also observed that the costs of compositions, in Figs 9(a) and (b), do not change supporting the fact that using VOC^E does not affect compositions.

Our experiments provide two conclusions: First, by augmenting WSC with VOC based 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 average overall cost of the WSC when adapted is significantly less than utilizing a non-changing policy. Second, we demonstrated that if service providers are able to provide longer guarantees on their service parameters, less computational effort needs to be spent on adapting the WSCs. This substantiates the intuition that in less volatile environments as formalized by higher expiration times, less adaptation is required to keep the WSC optimal.

X. DISCUSSION

Real world process environments are volatile – parameters of the services such as costs and availability may vary over time. In such environments, WSCs must adapt to the revised information to remain cost-effective.

We introduced a method, called the value of changed information, that queries and adapts a WSC to changes in

parameters of the services, only if the revised parameters are worth obtaining. Using two example scenarios, we show that our approach results in WSCs that are more cost-effective than approaches that do not adapt the composition or those that use naive querying strategies.

Furthermore, we mitigated the complexity of computing VOC so that this method of adaptation becomes feasible. We identified those parameter values that are not expected to cause changes in the composition and may be ignored while computing the VOC. We then exploited service parameter guarantees during which parameters’ values remain unchanged. Services whose parameter guarantees have not expired, need not be considered for querying. Using the scenarios, we empirically demonstrated the speedups when using these techniques.

The focus of our future work will be to improve upon our current adaptation method driven by VOC. Building a formal model of the volatility of a composition environment 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 ways of introducing new information to the model, when applicable, so as to avoid full re-computations of the policies.

ACKNOWLEDGMENT

This research is supported in part by grant number R01HL087795 from the National Heart, Lung, And Blood Institute. This content is solely the responsibility of the authors and does not necessarily represent the official views of the National Heart, Lung, And Blood Institute or the National Institutes of Health.

REFERENCES

- [1] J. Harney and P. Doshi. Adaptive web processes using value of changed information. In *ICSOC*, pages 179–190, 2006.
- [2] J. Harney and P. Doshi. Speeding up adaptation of web service compositions using expiration times. In *WWW*, pages 1023–1032, 2007.
- [3] W. M. P. van der Aalst and S. Jablonski, “Dealing with workflow change: identification of issues and solutions,” *IJCSSE*, vol. 15, no. 5, pp. 267–276, September 2000.
- [4] A. S. Y. Han and C. Bussler, “A Taxonomy of Adaptive Workflow Management,” in *CSCW-98 Workshop, Towards Adaptive Workflow Systems*, 1998.
- [5] A. Borgida and T. Murata, “Tolerating exceptions in workflows: a unified framework for data and processes,” in *WACC*, 1999, pp. 59–68.
- [6] N. Desai, A. K. Chopra, and M. P. Singh, “Business process adaptations via protocols,” in *SCC*, 2006, pp. 601–608.
- [7] D. M. Strong and S. M. Miller, “Exceptions and exception handling in computerized information processes,” *ACM Trans. Inf. Syst.*, vol. 13, no. 2, pp. 206–233, 1995.
- [8] M. Brambilla, S. Ceri, S. Comai, and C. Tziviskou, “Exception handling in workflow-driven web applications,” in *WWW*, 2005, pp. 170–179.
- [9] Z. Luo, A. P. Sheth, K. Kochut, and J. A. Miller, “Exception handling in workflow systems,” *Applied Intelligence*, vol. 13, no. 2, pp. 125–147, 2000.
- [10] R. Muller, U. Greiner, and E. Rahm, “Agentwork: a workflow system supporting rule-based workflow adaptation,” *Journal of Data and Knowledge Engineering*, vol. 51, no. 2, pp. 223–256, 2004.
- [11] N. C. Narendra and S. Gundugola, “Automated context-aware adaptation of web service executions,” in *AICCSA*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 179–187.
- [12] Z. Maamar, N. C. Narendra, D. Benslimane, and S. Sattanathan, “Policies for context-driven transactional web services,” in *CAiSE*, 2007, pp. 249–263.

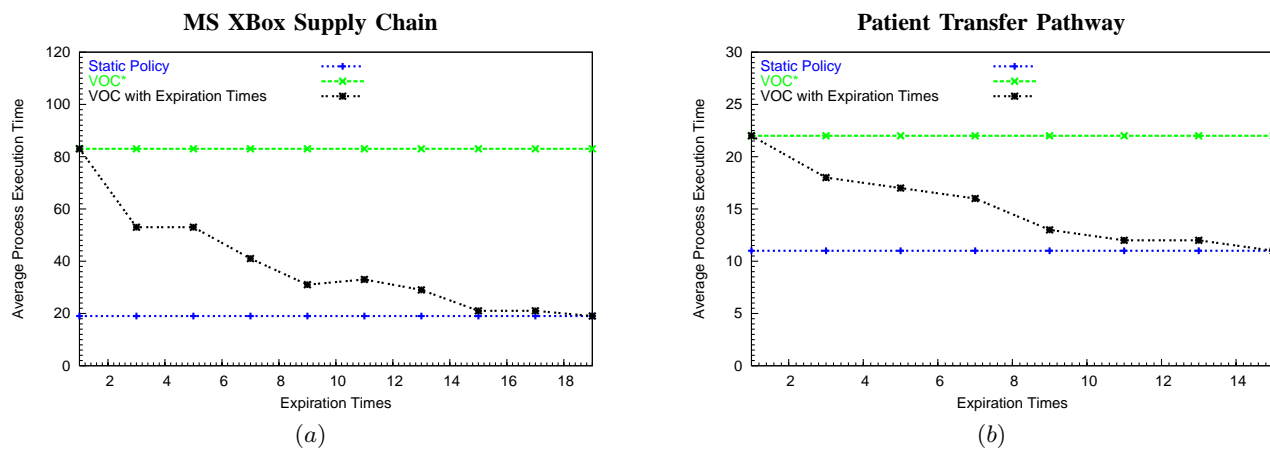


Fig. 16. (a) Average WSC execution times (in sec) when using the VOC^E based adaptation, VOC^* based adaptation and no adaptation. (b) shows analogous results for the patient transfer clinical pathway.

[13] M. Reichert and P. Dadam, "Adeptflex-supporting dynamic changes of workflows without losing control," *JHIS*, vol. 10, no. 2, pp. 93–17, 1998.

[14] E. A. Stohr and J. L. Zhao, "A technology adaptation model for business process automation," in *HICSS (4)*, 1997, pp. 405–414.

[15] W. M. P. V. D. Aalst, "Exterminating the dynamic change bug: A concrete approach to support workflow change," *Information Systems Frontiers*, vol. 3, no. 3, pp. 297–317, 2001.

[16] A. Sheth, J. Cardoso, J. Miller, and K. Kochut, "Qos for service-oriented middleware," in *SCI*, 2002, pp. 528–534.

[17] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava, "Adaptation in web service composition and execution," in *ICWS*, 2006, pp. 549–557.

[18] G. Chafle, P. Doshi, J. Harney, S. Mital, and B. Srivastava, "Improved adaptation of web service compositions using value of changed information," in *ICWS, Industry Track*, 2007, pp. 784–791.

[19] H. Paques, L. Liu, and C. Pu, "Adaptation space: A design framework for adaptive web services," *Int. J. Web Service Res.*, vol. 1, no. 3, pp. 1–24, 2004.

[20] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma, "Dynamic workflow composition using markov decision processes," *JWSR*, vol. 2, no. 1, pp. 1–17, 2005.

[21] T.-C. Au, D. S. Nau, and V. S. Subrahmanian, "Utilizing volatile external information during planning," in *ECAI*, 2004, pp. 647–651.

[22] T.-C. Au, U. Kuter, and D. S. Nau, "Web service composition with volatile information," in *ISWC*, 2005, pp. 52–66.

[23] T.-C. Au and D. Nau, "Reactive query policies: A formalism for planning with volatile external information," in *CIDM*, 2007, pp. 243–250.

[24] D. Gotz and K. Mayer-Patel, "A general framework for multidimensional adaption," in *ICME*, 2004, pp. 612–619–126.

[25] N. C. 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*, 2007, pp. 546–557.

[26] A. Charfi and M. Mezini, "Aspect-oriented web service composition with ao4bpel," in *ECOWS*, 2004, pp. 168–182.

[27] K. Gomadam, A. Ranabahu, L. Ramaswamy, A. P. Sheth, and K. Verma, "A semantic framework for identifying events in a service oriented architecture," in *ICWS*, 2007, pp. 545–552.

[28] K. Verma, P. Doshi, K. Gomadam, J. Miller, and A. Sheth, "Optimal adaptation in web processes with coordination constraints," in *ICWS*, 2006, pp. 257–264.

[29] Y. Wu and P. Doshi, "Regret-based decentralized adaptation of web processes with coordination constraints," in *SCC*, 2007, pp. 262–269.

[30] Microsoft, "Enabling an adaptable, aligned, and agile supply chain with biztalk server and rosettanet accelerator," Tech. Rep. <http://www.microsoft.com/technet/itshowcase/content/smbiztalktcs.msp>, 2005.

[31] V. Agarwal, G. Chafle, K. Dasgupta, N. M. Karnik, A. Kumar, S. Mittal, and B. Srivastava, "Synthy: A system for end to end composition of web services," *J. Web Sem.*, vol. 3, no. 4, pp. 311–339, 2005.

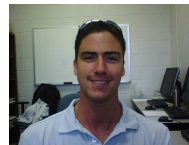
[32] M. L. Puterman, *Markov Decision Processes*. NY: John Wiley & Sons, 1994.

[33] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.

[34] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[35] B. Bojanov, H. Halcopian, and A. Sabakian, *Spline Functions and Multivariate Interpolations*. Kluwer Academic Publishers, 1993.

John Harney John Harney is currently a PhD candidate at the University of Georgia. His research interests lie in understanding volatility and its effect on service compositions, and intelligently adapting service compositions in response to revised information. He has published and presented papers in WWW, ICSOC and ICWS.



Prashant Doshi Dr. Prashant Doshi is an assistant professor of computer science at the University of Georgia and a member of the LSDIS lab. His research interests lie in both SOC, specifically in dynamic service compositions, and in AI, specifically in decision-making under uncertainty. He has carried out research in WS modeling, WS composition and constraint-aware adaptations of WS compositions. He has also had short stints at the IBM T. J. Watson Research Center where he worked in the eBusiness Group. He has published extensively in journals, conferences, and other forums in both the fields of SOC and AI. His research has led to publications in Journal of WS Research, WWW, ICWS, SCC, ICSOC, as well as Journal of AI Research, National Conf. on AI (AAAI), and AAMAS. He has served on the program committees of several conferences and workshops in the fields of SOC and AI.



APPENDIX 1 - DERIVATION OF THE GRADIENT DESCENT

Using the x-axis to represent the rate of order satisfaction \mathbf{p} and the y-axis to represent the error $\mathbf{E}(\mathbf{p})$ we may say:

$$\frac{\partial E(\mathbf{p})}{\partial \mathbf{p}} = \frac{\partial y}{\partial x}$$

Given end points P_0^j , where $P_{0,x}^j$ and $P_{0,y}^j$ is the x and y component of point 0 respectively of the j^{th} Bezier curve, and P_2^j and control point P_1^j , the equations for the quadratic Bezier curve are as follows:

$$\begin{aligned} x(\theta) &= (1-\theta)^2 P_{0,x}^j + 2\theta(1-\theta)P_{1,x}^j + \theta^2 P_{2,x}^j \\ y(\theta) &= (1-\theta)^2 P_{0,y}^j + 2\theta(1-\theta)P_{1,y}^j + \theta^2 P_{2,y}^j \end{aligned} \quad (12)$$

where $\theta \in [0,1]$ is a parameter. The equation for x may be rewritten as follows:

$$x(\theta) = P_{0,x}^j + (2P_{1,x}^j - 2P_{0,x}^j)\theta + (P_{0,x}^j + P_{2,x}^j - 2P_{1,x}^j)\theta^2$$

Let us focus on the final θ^2 term. Let the interval, $i = (P_{2,x}^j - P_{0,x}^j)$ and let points P_0^j , P_1^j and P_2^j be uniformly chosen over x . The term may be rewritten as:

$$2P_{0,x}^j + i - 2(P_{0,x}^j + i/2) = 0$$

We can therefore eliminate the θ^2 term and solve for θ :

$$\theta = -\frac{(P_{0,x}^j - x)}{2P_{1,x}^j - 2P_{0,x}^j}$$

We may now substitute this term for θ into $y(\theta)$ found in Eq. 12.

$$\begin{aligned} y &= \left(1 - \left(-\frac{(P_{0,x}^j - x)}{2P_{1,x}^j - 2P_{0,x}^j}\right)\right)^2 P_{0,y}^j + \left(-\frac{(P_{0,x}^j - x)}{2P_{1,x}^j - 2P_{0,x}^j}\right)^2 P_{2,y}^j \\ &\quad + 2\left(-\frac{(P_{0,x}^j - x)}{2P_{1,x}^j - 2P_{0,x}^j}\right)\left(1 - \left(-\frac{(P_{0,x}^j - x)}{2P_{1,x}^j - 2P_{0,x}^j}\right)\right)P_{1,y}^j \end{aligned}$$

Grouping together terms of x^2 , x , and a constant we obtain:

$$\begin{aligned} y &= \frac{P_{0,y}^j + P_{2,y}^j - P_{1,y}^j}{4(P_{1,x}^j)^2 - 8P_{1,x}^j P_{0,x}^j + 4(P_{0,x}^j)^2} x^2 \\ &\quad - \left(\frac{2P_{0,x}^j (P_{0,y}^j + P_{2,y}^j - 2P_{1,y}^j)}{4(P_{1,x}^j)^2 - 8P_{1,x}^j P_{0,x}^j + 4(P_{0,x}^j)^2} - \frac{P_{1,y}^j - P_{0,y}^j}{P_{1,x}^j - P_{0,x}^j}\right) x \\ &\quad + \left(P_{0,y}^j + \frac{(P_{0,x}^j)^2 (P_{0,y}^j + P_{2,y}^j - 2P_{1,y}^j)}{4(P_{1,x}^j)^2 - 8P_{1,x}^j P_{0,x}^j + 4(P_{0,x}^j)^2} - \frac{P_{0,x}^j (P_{1,y}^j - P_{0,y}^j)}{P_{1,x}^j - P_{0,x}^j}\right) \end{aligned}$$

Finally, we simply take the derivative of y with respect to x :

$$\begin{aligned} \frac{dy}{dx} &= 2 \frac{(P_{0,y}^j + P_{2,y}^j - 2P_{1,y}^j)}{4(P_{1,x}^j)^2 - 8P_{1,x}^j P_{0,x}^j + 4(P_{0,x}^j)^2} x \\ &\quad + \left(\frac{P_{1,y}^j - P_{0,y}^j}{P_{1,x}^j - P_{0,x}^j} - \frac{2P_{0,x}^j (P_{0,y}^j - P_{2,y}^j - 2P_{1,y}^j)}{4(P_{1,x}^j)^2 - 8P_{1,x}^j P_{0,x}^j + 4(P_{0,x}^j)^2}\right) \end{aligned}$$