

Selective Querying for Adapting Hierarchical Web Service Compositions

John Harney

and

Prashant Doshi

LSDIS Lab, Dept. of Computer Science,

University of Georgia,

Athens, GA 30602

1. INTRODUCTION

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

Dynamism manifests in WSC environments in a variety of ways. For example, changes range from the operational level (such as a newly introduced task) to the organizational level (such as new company policies) as mentioned in [van der Aalst and Jablonski 2000; Han and Bussler 1998]. Indeed, these surveys classify a variety of changes in different ways. Solutions have been presented to address some of these changes, ranging from exception handling techniques defined in [Borgida and Murata 1999] to instituting protocol adaptations in [Desai et al. 2006].

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

Author's email: jfh@cs.uga.edu

Author's email: pdoshi@cs.uga.edu

parameters of the participating services so as to optimize the composition.¹

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

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

Previously, we introduced a method to intelligently query for revised parameters using the *value of changed information (VOC)* [Harney and Doshi 2006]. In particular, we compute the trade-off between the cost of querying for up-to-date information² and the value of expected change in 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. In computing the VOC, we utilize stochastic models of volatility of each of the services’ parameters. We adopt a *myopic* approach in that we query only one service provider at a time and utilize the revised information for that WS, which leads to the maximum VOC.

While we previously demonstrated the usefulness of VOC in the context of simple WSCs [Harney and Doshi 2006], in this article, we expand the applicability of VOC to a hierarchical WSC. In many cases, to promote scalability a WSC may be seen as nested – a higher level WSC may be composed of WSs and lower level WSCs – which induces a natural hierarchy over the composition. In contrast to flat WSCs, a hierarchical decomposition introduces multiple challenges: (a) Because only the parameters of the lower level component services are known, we must derive the parameters of the composite service from these, and (b) we must derive a model of volatility for the composite service’s parameters from the models of its component services. While approaches for aggregating parameters of component WSs exist [Zhao and Doshi 2007; Cardoso et al. 2004], we present a way of formulating the stochastic models of volatility of composite services. Given approaches that address both these challenges, we show how we may compose and adapt hierarchical WSCs by descending down the levels of nesting and computing the VOC for WSCs at each level. This procedure is further complicated if we decide to query a composite service.

We show that, though myopic, our approach performs well in the presence of

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.

service data volatility, with reduced time and cost overhead caused by querying. In particular, our experiments demonstrate that the VOC mechanism avoids “unnecessary” queries in comparison to naive approaches of querying, say periodically. This translates to savings in overall costs for the WSC. For the purpose of evaluation, we utilize a realistic scenario of mortgage loan processing. Within our services-oriented architecture (SOA), we represent the mortgage broker’s WSCs using WS-BPEL [IBM 2005], and the different provider services as well as a service for computing the VOC using WSDL [WSDL Specification 2001].

This paper is structured in the following manner. We discuss related work in Section 2. In Section 3, we introduce our motivating scenario in detail. In Section 4, we describe the stochastic framework we use to formulate and execute our WSC. Additionally, we review the concept and definition of VOC and briefly describe how it may be used. We present its formulation in the context of a hierarchical WSC in Section 5 and introduce a recursive algorithm for composing and adapting a hierarchical WSC using VOC. In Section 6, we provide comparative experimental results. Sections 7 and 8 conclude the article with avenues for future work.

2. RELATED WORK

Dynamism manifests in WSC environments in a variety of ways [Han and Bussler 1998]. Our work focuses on data volatility, which van der Aalst describes as dynamism in the information perspective [van der Aalst and Jablonski 2000]. Dealing with changes of this type constitute only a small portion of the general adaptation problem. Indeed, research into adaptation has been broad and encompasses many different types of volatility.

Earlier work focused on handling exceptions that occur in workflows [Strong and Miller 1995; Brambilla et al. 2005; Luo et al. 2000]. 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 [Chiu et al. 1999; Muller et al. 2004]. Typically transaction constructs are employed such as task rollbacks and compensations [Narendra and Gundugola 2006; Maamar et al. 2007] to maintain correctness and consistent workflows [Reichert and Dadam 1998; Borgida and Murata 1999]. However, changes to the task instance data was often overlooked in favor of guaranteeing that tasks interoperate.

Research in volatility in other aspects of the WSC has also received attention. Stohr and Zhao [Stohr and Zhao 1997] focus on changes in the organizational perspective. They devised a Business Process Adaptation Model, which seeks to decide how changes in business technologies may affect the needs of business process automation. Here, they focus on choosing new technologies that benefit the general needs of an organization. Desai et al. [Desai et al. 2006; Desai et al. 2006] focus on adapting processes using handcrafted protocols. These business process protocols (set of rules that govern a business interaction) were created primarily to alleviate problems of heterogeneity and to support autonomy among different WS providers. Protocols may be changed to adapt to processes that require frequent changes in participating providers’ heterogeneous business models. Van der Aalst

et al. [Aalst 2001] addressed the problem of “dynamic change” in workflows. The dynamic change problem finds solutions to handling old instances in new workflow process definitions. The dynamic change problem is not relevant to this paper, as we only consider adapting compositions that are executed on an individual, case by case basis.

Only recently have researchers turned their efforts toward identification of change in the individual services’ parameters [Sheth et al. 2002]. In [Chafle et al. 2006; Chafle et al. 2007] 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 may not be used making the approach inefficient. Further, there is no guarantee of optimality of the resulting WSCs. Paques et al. [Paques et al. 2004] 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. [Doshi et al. 2005] 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 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 (process flow) re-computations that do not bring about any change in the WSC. Au et al. [Au et al. 2004; Au et al. 2005; Au and Nau 2007] introduce a framework that composes processes in the presence of data volatility. Using a reactive querying policy, they obtain current parameters of the WSC by querying WS providers only when the parameters expire. While this is similar in concept to our approach, plan re-computation is assumed to take place irrespective of whether the revised parameter values are expected to bring about a change in the composition. This may lead to frequent unnecessary computations. Gotz and Mayer-Patel [Gotz and Mayer-Patel 2004] incorporate multidimensional data adaptation using a metric similar to the value of information to determine if new information may impact the utility of their application. The authors, however, primarily focus on multimedia applications rather than business processes.

Nanjangud et al. [Narendra et al. 2007] and Charfi et al. [Charfi and Mezini 2004] apply the aspect-oriented programming idea to WSCs. Aspects were used to adapt to changes in WS components dynamically and consistently. Analogous to the traditional exception-handling techniques, this line of work focuses on composition correctness and consistency. Gomadam et al. [Gomadam et al. 2007] 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. [Verma et al. 2006] and Wu and Doshi [Wu and Doshi 2007] 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.

3. MOTIVATING SCENARIO: MORTGAGE LOAN PROCESSING

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

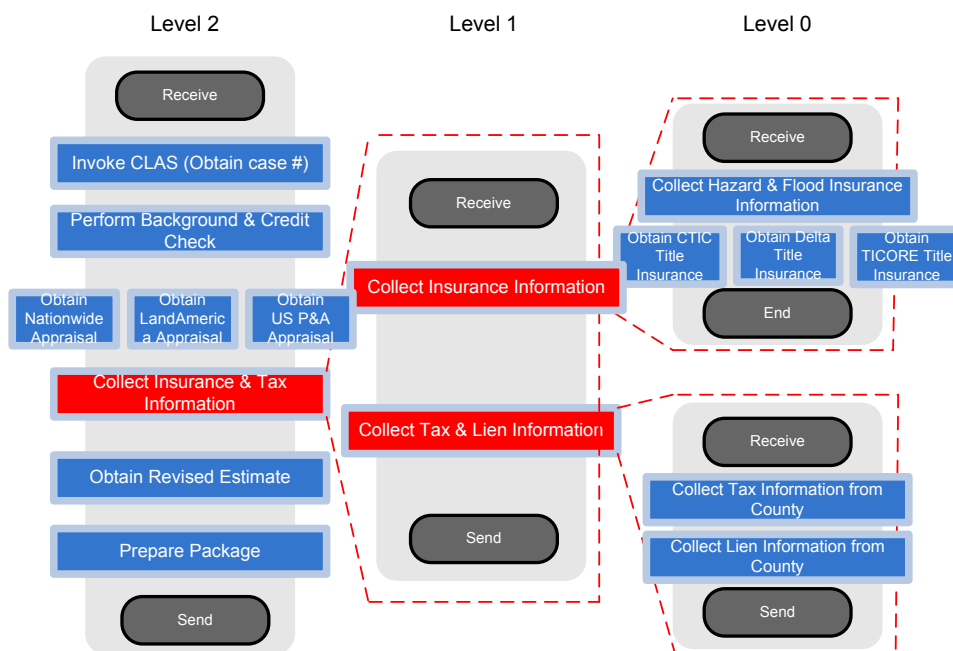


Fig. 1. A hierarchical mortgage loan process used by a mortgage broker. The broker may choose among multiple appraisal companies and title insurance companies at different levels in the composition.

Figure 1 illustrates an example process utilized by the broker. We begin with a description of the upper level of the hierarchy (labeled “Level 2”). The broker receives a request from a client interested in securing a mortgage loan. Some of the activities that the broker performs are to issue a request to the CLAS (CHUMS Lender Access System) WS to obtain a case number for this particular client, utilize the WS of the credit rating agencies to perform financial background and credit checks of the client, and hire an appraisal agency to appraise the value of the home. In this example, the broker may choose between three different appraisal services: Nationwide, LandAmerica, and US P and A. A choice among these is made depending on the cost and the availability of using the service at a given time. If a chosen service is unable to perform the appraisal, then one of the other appraisal services may be used instead. Next, the broker invokes the insurance and tax information collection service to obtain vital home insurance, tax and lien information. With

all of this information on hand, the broker will complete an estimate of the loan and prepare the complete package for the client to review.

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

Therefore, the example reveals two important factors for choosing optimally. First, the broker must know the certainty with which the appraisal requests will be satisfied by each of the agencies. Second, at each stage, rather than greedily selecting an action with the least cost, the broker must select the action which is expected to be optimal over the long term.

4. BACKGROUND

For the purpose of illustration, we select a decision-theoretic planning technique for composing WSs [Doshi et al. 2005]. However, our approach is applicable to any model based service composition technique such as [Agarwal et al. 2005] (for example, see [Chafle et al. 2007]). We then briefly review the definition and formulation of the *value of changed information* (VOC) of participating WSs, and refer the reader to [Harney and Doshi 2006] for more details.

4.1 Web Service Composition Using MDP

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

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

where $S = \prod_{i=1}^n X^i$, S is the set of all possible states factored into a set, X , of n variables, $X = \{X^1, X^2, \dots, X^n\}$; A is the set of all possible actions; T is a transition function, $T : S \times A \rightarrow \Delta(S)$, which specifies the probability 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 problem of composing the component WSs of **Insurance Information** composite service at level 1 of the mortgage scenario in Fig. 1 as a MDP. The state of the composition is captured by the random variables – **hazard and flood insurance information available**, **CTIC title insurance available**, **Delta insurance available**, and **TICORE title insurance available**. A state is then a conjunction of assignments of either *Yes*, *No*, or *Unknown* to each random variable. Actions are WS invocations,

$A = \{\text{Hazard and Flood Insurance Information, CTIC Title Insurance, Delta Insurance, and TICORE Title Insurance}\}$.

The transition function, T , models the non-deterministic effect of each action on some random variable(s). For example, invoking the WS CCTIC Title Insurance will cause **CTIC title insurance available** to be assigned *Yes* with a probability of $T(\text{CTIC title insurance available} = \text{Yes} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown})$. This rate of order satisfaction depends on two probabilities: (1) the probability that CTIC is able to provide the insurance, and (2) the availability of the CTIC Title Insurance's WS interface. If the two availabilities are independent of one another ³, we may view T as a product of these two probabilities: $T(\text{CTIC title insurance available} = \text{Yes} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = Pr(\text{CTIC title insurance product availability} = \text{Yes}) \times \text{WS Availability}$. Similarly, the **CTIC title insurance availability** will be assigned *No* with a probability of $T(\text{CTIC title insurance available} = \text{No} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = Pr(\text{CTIC title insurance product availability} = \text{No}) \times \text{WS Availability}$, and *Unknown* with a probability of $T(\text{CTIC title insurance available} = \text{Unknown} | \text{CTIC Title Insurance, CTIC title insurance available} = \text{Unknown}) = 1 - \text{WS Availability}$. 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 CTIC Title Insurance WS and the cost of the insurance itself. We let H be some finite value which implies that the broker is concerned with getting the most optimal WSC possible within a fixed number of steps. Since no information is available at the start state, all random variables will be assigned the value *Unknown*.

Once the mortgage broker has modeled its WS composition problem as a MDP, it may apply standard MDP solution techniques to arrive at an optimal process. These solution techniques revolve around the use of stochastic dynamic programming [Puterman 1994] for calculation of the optimal policy using *value iteration*:

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

where:

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

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

Once we know the expected cost associated with each state of the process, the

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

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

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

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

Algorithm for generating WSC

Input: π^*, s_0

$s \leftarrow s_0$

while goal state not reached

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

Execute Web service a

Get response of a and construct next state, s'

$s \leftarrow s'$

end while

end algorithm

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

The reader at this point may wonder that how does an optimal policy such as π^* translate to an optimal composition. In Fig. 2, we give an algorithm that addresses this question. It takes the optimal policy, and the starting state of the composition as input, and interleaves composition and execution of the process. 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 is integrated into the random variable set, effectively transitioning into a new state. This process is repeated until the desired goal state is reached.

4.2 Value of Changed Information

As discussed previously, the parameters of the participating services and providers may change during the life-cycle of a WSC. For example, the cost of using the Delta Title Insurance provider's services may increase or the probability with which Delta can process a request may reduce. The former requires an update of the cost function, C , while the latter requires an update of the transition function, T , in the MDP model. In this article, we focus on a change in the transition function T , though our approach is generalizable to fluctuations in other model parameters as well.

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

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

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 using an example below:

Example 1. As a simple illustration, let us suppose that we intend to query the **CTIC Title Insurance WS** provider for its current rate of order satisfaction. Eq. 4 becomes,

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

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

Let $V_{\pi}(s|T')$ be the expected cost of following the original policy, π , from 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) [Russell and Norvig 2003]. Indeed, both of them may be seen as special cases of the value of information idea, which determines whether new information is useful to a particular process. However, there is an important difference between the two concepts. VPI computes the value of *additional* information, while the VOC provides the value of *revised* information. We illustrate this distinction using an example:

Example 2. In the mortgage acquisition process, the VPI provides a way to gauge the expected impact of knowing additional (previously unknown) parameters of Ws 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 request satisfaction rate and service cost.

Analogous to VPI, the following theorem holds for VOC, whose proof is in [Harney and Doshi 2006].

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

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

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

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

5. HIERARCHICAL WEB SERVICE COMPOSITION WITH VOC

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

In [Zhao and Doshi 2007], Zhao and Doshi provide a method of composition that exploits a hierarchical decomposition. We slightly modify this approach and model each level of the hierarchy using a MDP. Specifically, the lowest levels of the hierarchy (leaves) are modeled using a MDP containing *primitive* actions, which are invocations of the Ws. Higher levels of the composition problem are modeled using MDPs that contain *abstract* actions, which represent the execution of lower level WSCs. While formulating the lowest level MDPs is straightforward and proceeds as in Section 4.1, we must derive the parameters of the composite WS to permit

the formulation of the MDP that models the composition problem at the higher level WSCs.

5.1 Model Parameters for Composite Web Services

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

Zhao and Doshi [Zhao and Doshi 2007] utilize the correspondence between the high level abstract actions and the corresponding low level primitive actions. Let the abstract action, \bar{a} , represent the sequential execution, in some order, of primitive actions, $\{a_1, a_2\}$, of the underlying primitive MDP. Because the order in which the primitive actions a_1 and a_2 are performed is not known from beforehand, there may be multiple ways to achieve the composition – start from the state s_p and reach the state, s_e . Let $s_p \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_e$ be one such path, where s_1 is an intermediate state of the WSC, then, $T(s_1|a_1, s_p) \times T(s_e|a_2, s_1)$ is the probability of following this path, where T is the transition function of the primitive MDP. The required probability, $Pr(s_e|s_p)$, is the sum of the probabilities of following all such paths. Analogously, the cost of performing the abstract action is the average of the cost of following each of the possible paths that achieve the composition weighted by the probability of that path.

We may model the mortgage broker’s beliefs over the possible parameters of the WS, ($Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 5) using density functions. We let the densities for the WSs take the form of *Gaussian* density functions⁴. Fig 3 shows examples of such densities, defined for the **Hazard and Flood Insurance Information** WS (Fig 3(a)) and the title insurance WSs (Fig 3(b)). Other WSs in the mortgage loan process are assumed to have analogous densities. We emphasize that these densities are marginalizations of the more complex ones that would account for all the factors that may influence, for example, a service’s rate of request satisfaction.

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

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

⁴Note that other density functions (such as betas and polynomials) may also be used

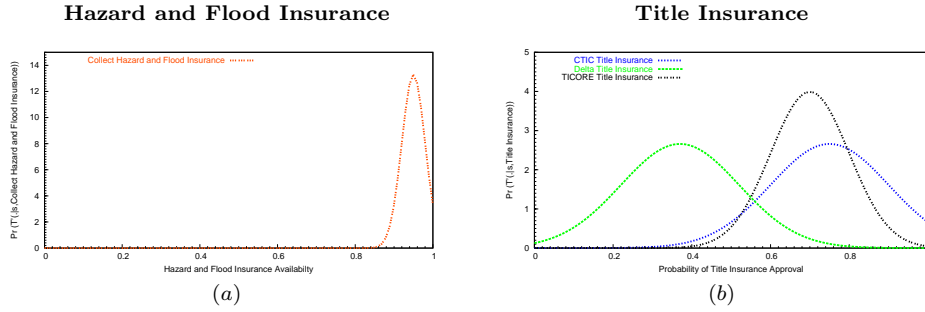


Fig. 3. Example probability density functions representing the mortgage broker's beliefs over the (a) Hazard and Flood Insurance WS and (b) Title Insurance WSs' probabilities of satisfying requests.

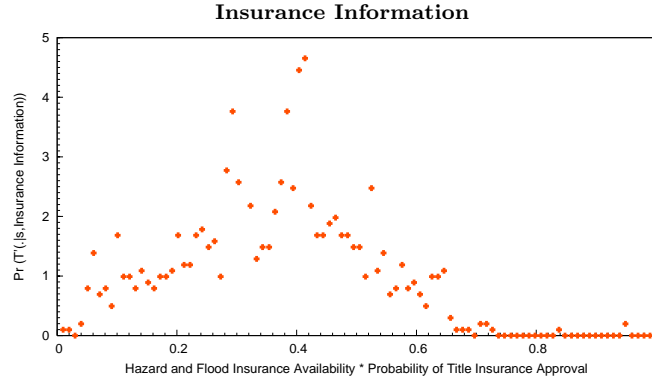


Fig. 4. The approximate probability density function representing the mortgage broker's beliefs over the composite service **Insurance Information**. The density is over a product space composed of the transition probabilities of the actions that represent the lower level WSs (Hazard and Flood Insurance and Delta Insurance).

Therefore, the belief density over the availability of Insurance Information, will be the summation of functions over the product space: availability of Hazard and Flood Insurance WS \times availability of a Title Insurance service.

Although we model the densities over availability of individual WSs as Gaussians (for example Figs. 3(a) and (b)), the function over the product space is not a Gaussian but rather a modified Bessel function of the second kind [Glen et al. 2004]. Because generating the Bessel function exactly is complex, we utilize a sampling scheme to generate the density over the product space, which converges to the exact as the number of samples approaches infinity. We show an approximate density obtained by the sampling scheme in Fig. 4. Densities analogous to this one will be summed to obtain the broker's belief over the aggregate parameter of the composite WS **Insurance Information**.

5.2 Algorithm

We need only to make a small change to the algorithm outlined in Fig 2 to utilize VOC. In order to formulate and execute the WSC, we simply look up the current

```

Algorithm for adaptive Web service compositions – AWSC
Input:
   $\pi^*$  //optimal policy
   $s_0$  //initial state
   $l$  //depth

   $s \leftarrow s_0$ 
  while  $n > 0$ 
    if  $VOC^*(s) > QueryCost(T'(\cdot|a, s'))$  then
      if  $a^*$  is composite
         $a_k^* \leftarrow findA^*(l - 1, A, s)$ 
        Query  $a_k^*$ 
         $T' \leftarrow FormNewTransition(Level\ l, Level\ k)$ 
        Calculate policy  $\pi_n^*$  using the new MDP with  $T'$ 
       $a \leftarrow \pi_n^*(s)$ 
      if  $a$  is composite
        Recursively call AWSC algorithm for composite service  $a$ 
      else
        Execute primitive Web service  $a$ 
      Get response of  $a$  and construct next state  $s'$ 
       $s \leftarrow s'$ 
       $n \leftarrow n - 1$ 
    end if
  end algorithm

```

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

state of the WSC in the policy and execute the WS prescribed by the policy for that state. The response to the WS invocation determines the next state of the WSC. We adapt the composition to consider fluctuations in the model parameters by interleaving the formulation with VOC based selective querying.

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

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

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

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

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

After querying a_k^* , where k represents the level at which the queried WS resides, we must formulate a new transition, T' , and policy, π^* , for the level k WSC.

```

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

     $a_k^* \leftarrow \underset{a \in A}{\operatorname{argmax}} \operatorname{VOC}_{T(\cdot|a, s^k)}(s_k)$ 
    if  $a_k^*$  is a primitive service then
      return  $a_k^*$ 
    else //  $a_k^*$  is a composite service
       $s^{k-1} \leftarrow$  initial state of the process at level  $k - 1$ 
      return  $\operatorname{findA}^*(k - 1, A, s^{k-1})$ 
  end algorithm

```

Fig. 6. Function findA* recursively finds a WS that yields the highest VOC.

```

Algorithm for FormNewTransition(Level  $l$ , Level  $k$ )
  Input :
     $k$  //depth
     $l$  //current level

    if  $l > k$  then
       $T' \leftarrow \operatorname{FormNewTransition}(l - 1, k)$ 
      Formulate new  $T_c$  for composite WS given  $T'$ 
      Calculate new policy  $\pi'$ 
      return  $T_c$ 
    else
      Integrate revised information from query to form  $T'$ 
      Calculate new policy  $\pi'$ 
      return  $T'$ 
  end algorithm

```

Fig. 7. Recursively update the transition function and policies in the hierarchical composition.

Subsequently, new transition function (for the corresponding composite WS) and policies must be computed at all levels up to the top most level, l . For example, if the CTIC Title Insurance is being queried, we reformulate the policy at level 0 given the revised information, and recompute the aggregate parameters of the composite WS, Insurance Information at level 1. We subsequently revise the transition function, T' and resolve π^* at level 1. This recursive procedure is presented in Fig. 7.

6. EXPERIMENTAL RESULTS

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

6.1 Architecture

The algorithm described in Fig. 5 is implemented as a WS-BPEL [IBM 2005] flow and all WSs were implemented using WSDL [WSDL Specification 2001].

BPEL Implementation of Mortgage Acquisition Process

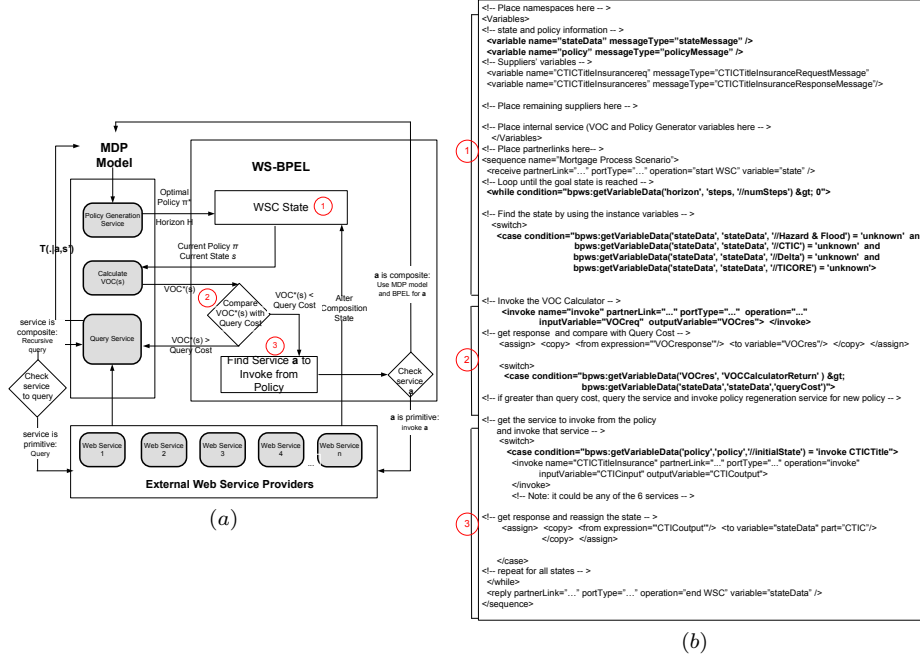


Fig. 8. SOA for implementing our adaptive WSC. (a) demonstrates the interaction of the composition with internal services. In (b), we show a sample of the BPEL markup for the mortgage acquisition process. Labels (1) (2) and (3) in (a) correspond to its associated markup in (b).

To the WS-BPEL flow, we give the optimal policy, π^* , of the top level WSC, the start state, and horizon as input. Our experiments utilized IBM's BPWS4J engine for executing the BPEL process and AXIS 2.0 as the container for the WSs. We show our SOA in Fig. 8(a).

Within our SOA, we provide internal WSs for solving the MDP model of the composition problem and generating the policy, and for computing the VOC. If the VOC exceeds the cost of querying a particular service provider (this cost is also provided as an input), the WS-BPEL flow invokes a special WS whose function is to query the service provider's information-providing WSs for revised information. This information is used to formulate and solve a new MDP and the output policy is fed back to the WS-BPEL flow. This policy is used by the WS-BPEL flow to invoke the prescribed external WS if it is not a composite one, and the response is used to formulate the next state of the process. If the WS to invoke is composite, the procedure is recursively repeated for the lower level WSC. This procedure continues until the goal state is reached or the total number of steps are exhausted.

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

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

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

The $\langle \textit{while} \rangle$ condition corresponds to the while loop in Fig. 5. Each state has an associated $\langle \textit{switch} \rangle \langle \textit{case} \rangle$ construct. In each $\langle \textit{case} \rangle$, the WSC invokes the compute 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 policy is then used to recommend the optimal service to invoke in the state. This process repeats until the composition has terminated (i.e. all steps have been exhausted).

6.2 Performance Evaluation

We utilized the mortgage loan processing scenario (Section 3) for our evaluations. We simulated querying the different WSs for their current percentage of request satisfaction (availability).⁵

We model the mortgage broker’s beliefs over the possible parameters of the individual WSs, ($Pr(T'(\cdot|a, s') = \mathbf{p})$ in Eq. 5) using *Gaussian* density functions. For composite WSs, we derive the densities over the aggregated parameters as shown in Section 5.1.

In Fig. 9 we compare the VOC-driven selective querying with four other strategies with respect to the average cost incurred from the execution of the adapted hierarchical WSCs, as the cost of querying the WSs for information is increased. Our methodology consisted of running a trial of 150 independent instances of each composition within a simulated volatile environment, where the queried parameters of the services were distributed according to the corresponding density plots (see Figs. 3 and 4 for example). We ensured that the compositions using each of the five strategies received similar responses from the services.

We utilized the following four other approaches for adaptation:

- (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.

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

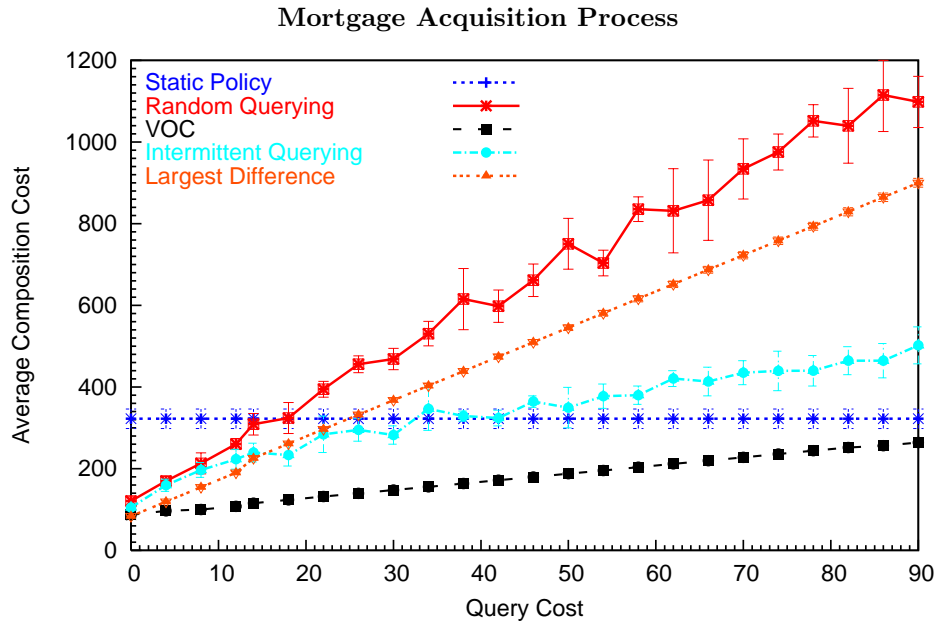


Fig. 9. Comparisons of the VOC based adaptive WSC with the static policy and other querying approaches for the mortgage loan acquisition. Lower average cost indicates better performance.

- (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 and as the costs of querying increase, we reduce the frequency with which we query services.
- (4) **Largest difference** This approach utilizes the distributions of the services parameters shown in Fig. 3. It selects a service to query whose existing parameter value is most different from the mean as obtained from the corresponding distribution.

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

Intuitively, as we increase the cost of querying, the VOC based approach performs less queries and adapts the WSC less. For large query costs, its performance is similar to using a WSC with an unchanging policy. For smaller query costs, a VOC based approach will query frequently, though not as much as a strategy that always queries 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, though worse than the VOC based

approach. This is because the service exhibiting the largest difference from the mean in its parameter value is often the one that brings about the largest change in the composition. However, this is not always the case – for example, a large change in the parameter of a mandatory service, such as the **Credit Check** service in the mortgage process, does not affect the composition though the approach will query it and incur the query cost. In addition, the difference in parameter values is not comparable to query costs. In summary, a WSC that is adapted using VOC performs better (incurs less average cost) because only significant changes to the WSC are carried out while simultaneously avoiding frequent costly queries.

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

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

We point out that adaptation using VOC comes at a computational price. In Table. I we give the run times for each of the comparative strategies used previously. We first observe that an adaptive WSC that uses VOC runs two to three times slower than a WSC that does not adapt (static policy). However, this time difference reduces when other strategies for adapting WSCs are utilized. The additional runtime of the VOC is primarily due to the computations required for Eq. 5. All of the adaptive approaches suffer from a lag time in receiving the query response from the WSs. Notice, however, that as the query cost increases, the time required by the VOC based approach decreases, because it issues less queries.

7. FUTURE TRENDS

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

8. CONCLUSION

Real world process environments are volatile – parameters of the services and providers such as costs and reliability may vary over time. In such environments, WSCs must adapt to the revised information to remain cost-effective. Previously, the value of changed information, has been used to gauge the value of the expected change that revised information may bring to the WSC, and it is compared with the cost of obtaining the information. If the probable revised information is worth the cost of obtaining it, the providers are queried for their WS’s current parameters

and we reformulate the WSC using the revised information. While previously, the VOC was applied toward adapting simple flat WSCs, in this article, we extend its applicability to hierarchical WSCs. Two of the primary challenges that we address are how to obtain beliefs over volatility of composite WS parameters, and which of the component WSs to invoke if a composite WS is found to potentially cause the most expected change in the process.

REFERENCES

- AALST, W. M. P. V. D. 2001. Exterminating the dynamic change bug: A concrete approach to support workflow change. *Information Systems Frontiers* 3, 3, 297–317.
- AGARWAL, V., CHAFLE, G., DASGUPTA, K., KAMIK, N., KUMAR, A., MITTAL, S., AND SRIVASTAVA, B. 2005. Synth: A system for end to end composition of web services. In *Journal of Web Semantics*. Vol. 3.
- AU, T.-C., KUTER, U., AND NAU, D. S. 2005. Web service composition with volatile information. In *International Semantic Web Conference (ISWC)*. 52–66.
- AU, T.-C. AND NAU, D. 2007. Reactive query policies: A formalism for planning with volatile external information. In *CIDM*.
- AU, T.-C., NAU, D. S., AND SUBRAHMANIAN, V. S. 2004. Utilizing volatile external information during planning. In *European Conference on Artificial Intelligence (ECAI)*. 647–651.
- BORGIDA, A. AND MURATA, T. 1999. Tolerating exceptions in workflows: a unified framework for data and processes. In *WACC*. 59–68.
- BRAMBILLA, M., CERI, S., COMAI, S., AND TZIVISKOU, C. 2005. Exception handling in workflow-driven web applications. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*. ACM, New York, NY, USA, 170–179.
- CARDOSO, J., SHETH, A., MILLER, J., ARNOLD, J., AND KOCHUT, K. 2004. Quality of service for workflows and web service processes. *Journal of Web Semantics* 1.
- CHAFLE, G., DASGUPTA, K., KUMAR, A., MITTAL, S., AND SRIVASTAVA, B. 2006. Adaptation in web service composition and execution. In *International Conference on Web Services (ICWS), Industry Track*.
- CHAFLE, G., DOSHI, P., HARNEY, J., MITAL, S., AND SRIVASTAVA, B. 2007. Improved adaptation of web service compositions using value of changed information. In *International Conference on Web Services (ICWS)*.
- CHARFI, A. AND MEZINI, M. 2004. Aspect-oriented web service composition with ao4bpel. In *European Conference on Web Services (ECOWS)*. 168–182.
- CHIU, D. K. W., LI, Q., AND KARLAPALEM, K. 1999. A meta modeling approach to workflow management systems supporting exception handling. *Information Systems* 24, 2, 159–184.
- DESAI, N., CHOPRA, A. K., AND SINGH, M. P. 2006. Business process adaptations via protocols. In *IEEE International Conference on Services Computing (SCC'06)*. 601–608.
- DESAI, N., MALLYA, A. U., CHOPRA, A. K., AND SINGH, M. P. 2006. Owl-p: A methodology for business process modeling and enactment. In *Agent-Oriented Information Systems-III, Lecture Notes in Computer Science (LNCS), Volume 3529*. 79–94.
- DOSHI, P., GOODWIN, R., AKKIRAJU, R., AND VERMA, K. 2005. Dynamic workflow composition using markov decision processes. *Journal of Web Services Research (JWSR)* 2, 1, 1–17.
- GLEN, A. G., LEEMIS, L., AND DREW, J. H. 2004. Computing the distribution of the product of two continuous random variables. *Computational Statistics & Data Analysis* 44, 3, 451–464.
- GOMADAM, K., RANABAHU, A., RAMASWAMY, L., SHETH, A. P., AND VERMA, K. 2007. A semantic framework for identifying events in a service oriented architecture. In *International Conference on Web Services (ICWS)*.
- GOTZ, D. AND MAYER-PATEL, K. 2004. A general framework for multidimensional adaption. In *ICME*. 612–619–126.
- HAN, A. S. Y. AND BUSSLER, C. 1998. A Taxonomy of Adaptive Workflow Management. In *CSCW-98 Workshop, Towards Adaptive Workflow Systems*.

- HARNEY, J. AND DOSHI, P. 2006. Adaptive web processes using value of changed information. In *International Conference on Service-Oriented Computing (ICSOC)*.
- IBM. 2005. *Business Process Execution Language for Web Services version 1.1*.
- LUO, Z., SHETH, A. P., KOCHUT, K., AND MILLER, J. A. 2000. Exception handling in workflow systems. *Applied Intelligence* 13, 2, 125–147.
- MAAMAR, Z., NARENDRA, N. C., BENSLIMANE, D., AND SATTANATHAN, S. 2007. Policies for context-driven transactional web services. In *CAiSE*. 249–263.
- MULLER, R., GREINER, U., AND RAHM, E. 2004. Agentwork: a workflow system supporting rule-based workflow adaptation. *Journal of Data and Knowledge Engineering*. 51, 2, 223–256.
- NARENDRA, N. C. AND GUNDUGOLA, S. 2006. Automated context-aware adaptation of web service executions. In *AICCSA '06: Proceedings of the IEEE International Conference on Computer Systems and Applications*. IEEE Computer Society, Washington, DC, USA, 179–187.
- NARENDRA, N. C., PONNALAGU, K., KRISHNAMURTHY, J., AND RAMKUMAR, R. 2007. Run-time adaptation of non-functional properties of composite web services using aspect-oriented programming. In *International Conference on Services Oriented Computing (ICSOC)*. 546–557.
- PAQUES, H., LIU, L., AND PU, C. 2004. Adaptation space: A design framework for adaptive web services. *International Journal of Web Service Research* 1, 3, 1–24.
- PUTERMAN, M. L. 1994. *Markov Decision Processes*. John Wiley & Sons, NY.
- REICHERT, M. AND DADAM, P. 1998. Adeptflex-supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems* 10, 2, 93–17.
- RUSSELL, S. AND NORVIG, P. 2003. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall.
- SHETH, A., CARDOSO, J., MILLER, J., AND KOCHUT, K. 2002. Qos for service-oriented middleware.
- STOHR, E. AND ZHAO, J. 1997. A technology adaptation model for business process automation.
- STRONG, D. M. AND MILLER, S. M. 1995. Exceptions and exception handling in computerized information processes. *ACM Trans. Inf. Syst.* 13, 2, 206–233.
- VAN DER AALST, W. M. P. AND JABLONSKI, S. 2000. Dealing with workflow change: identification of issues and solutions. *International Journal of Computer Systems Science and Engineering* 15, 5 (September), 267–276.
- VERMA, K., DOSHI, P., GOMADAM, K., MILLER, J., AND SHETH, A. 2006. Optimal adaptation in web processes with coordination constraints. In *International Conference on Web Services (ICWS)*.
- WSDL Specification 2001. *Web Services Description Language (WSDL) 1.1*.
- WU, Y. AND DOSHI, P. 2007. Regret-based decentralized adaptation of web processes with coordination constraints. In *IEEE SCC*. 262–269.
- ZHAO, H. AND DOSHI, P. 2007. Haley: A hierarchical framework for logical composition of web services. In *International Conference on Web Services (ICWS)*. 312–319.