

# Dynamically Managing the Delivery of Web Services via Workflow and Feedback

Randy Howard and Larry Kerschberg

George Mason University E-Center for E-Business, <http://eceb.gmu.edu/>  
[choward@gmu.edu](mailto:choward@gmu.edu), [kersch@gmu.edu](mailto:kersch@gmu.edu)

## Abstract

*The concept of automating the delivery of Web services functionality is an active research topic. This paper presents a comprehensive and overarching framework that supports the execution and delivery of Semantic Web services within the context of workflow management, and addresses the issues related to Virtual Organizations. The goal is to add semantics to Web services to endow them with capabilities currently lacking in the literature, but necessary for their successful deployment in future systems.*

*In this paper, we indicate how our Knowledge-based Dynamic Semantic Web Services (KDSWS) Framework addresses the activities involved in managing and monitoring Semantic Web Services by using both a meta-model and meta-process approach. In particular, the following issues are addressed: 1) semantic specification of services capabilities including quality of service, trust, and security; 2) the execution and delivery of services 3) workflow management; and 4) resource management, interoperation and evolution of the Virtual Organization.*

**Keywords:** Semantic Web Services, Virtual Organization, Service Delivery, Management, Workflow, Feedback

## 1. Introduction

The relatively new concept of *Web services* is important to both e-Business and e-Government in that applications may exchange functionality and information over the Internet. Web services standards provide XML-based protocols to find publicly-registered services, to understand their purpose and operation, to negotiate and agree upon usage charges and Quality-of-Service commitments, and to invoke the services within the context of Internet-based workflow coordination of these services.

“Semantic Web Services” (SWS) is the term that describes our research approach. We view “Web services” as services that use little or no semantic markup, and have little of the enhanced capability described in this paper. Semantic Web technology, on the other hand adds semantic and process oriented information, together with heuristics and constraints that can be used to coordinate the activities of the Virtual Organization (VO).

The dynamic delivery of functionality via Semantic Web Services involves managing issues that course through the enterprise such as transaction control, quality of service, security, interoperation, transportation. Additionally, the management of workflow and feedback is integral to this automation, and is the focus of this paper. This research places emphasis on this delivery to fulfill requests; thus, the term “fulfillment” is used to describe the delivery of Semantic Web Services.

Workflow was originally defined as “the computerized facilitation or automation of a business process, in whole or part”, and dealt only with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal [1]. However, with the ever-increasing dependence on the Internet, workflow management

has evolved into a network-centric discipline where process coordination must provide the means to improve the quality of service, increase flexibility, allow more choices, and support more complex services offered by independent service providers communicating asynchronously in an information grid [2]. (For more details how this research deals with grids, see[3].)

This research considers feedback to include delivering functionality to the requester, informing requestors about the status of that delivery, and measuring the performance results of delivery components in order to optimize the request fulfillment in the future. Feedback is bi-directional between the provider and the client because the provider takes on the role of clients and vice-versa. Although the ultimate goal is to fully automate Semantic Web Services, automation still requires human intervention at times.

This paper references the Knowledge-based Dynamic Semantic Web Services (KDSWS) Framework [4] that addresses, in an integrated end-to-end manner, the life-cycle of activities involved in preparing, publishing, requesting, discovering, selecting, configuring, deploying, and delivering Semantic Web Services. In particular, the following issues are addressed: 1) semantic specification of services capabilities including quality of service, trust, and security; 2) the execution and delivery of services 3) workflow management; and 4) resource management, interoperation and evolution of the Virtual Organization.

Section 1 has introduced the topic, while Section 2 discusses the problem space and some of the drawbacks and issues associated with existing approaches. The Knowledge-based Dynamic Semantic Web Services Framework is presented in Section 3. Section 4 presents a methodology to dynamically deliver Web services within the KDSWS framework. Section 5 presents our conclusions and suggests areas for future research.

## 2. Related Work

Semantic Web Services attempt to address the shortcomings of Web services with respect to automation, and OWL-S [5] is being established as a primary target specification for this endeavor. However, recent literature [6-8] has documented several shortcomings attributed to OWL-S. With respect to this research, the notable limitations are:

- **Conceptual ambiguity** in that the major concepts of OWL-S (service, grounding, and process) are still being clarified. For example, the concept of *service* is regarded as “any Web-accessible program/sensor/device” [7, 9].
- **Narrow scope** in that OWL-S also does not address fully the real-world issues [7] related to the VO. For example, OWL-S considers the service and the process aspects as the primary elements, but does not show how to specify federation, agent and data store aspects.
- **Loose design** means that OWL-S does not adequately differentiate the context of the service-related tasks such as discovery, composition and invocation [7].
- **Coupled function and description** in that OWL-S lacks an explicit and declarative decoupling between the functional features of a process (what) and the structural description of such processes (how) [6].
- **Static process declaration** means that OWL-S does not provide for the dynamic adjustment of an agent’s Process model during the interaction [8].

In order to deliver Web services, many disparate protocols and technologies (see those in [10]) are required to achieve an end-to-end solution, despite the fact that they are all based on an XML foundation. The nuances of these various technologies still need to be mediated to some degree to be truly interoperable [11].

Attempts have been made to enable the dynamic supply-chain reconfiguration with agent technology and dynamic workflow technology. A large body of work (e.g., ARIS [12], BPEL4WS [13], Business Process Modeling Language (BPML) [14]) are beginning to be deployed into industry and real applications. The Web Services Choreography Description Language (WS-CDL) specification was recently released to compose interoperable peer-to-peer collaborations between Web service participants. WS-CDL produces a ‘global’ definition of the “common ordering conditions and constraints under which messages are exchanged” in order to describe a “common and complementary observable behavior of all the Web Services participants involved” [15]. Some elements of WS-CDL’s model parallel the elements of this research’s modeling specifications.

The vision of ebXML (Electronic Business using eXtensible Markup Language) is “to create a single global electronic marketplace where businesses can find each other, agree to become trading partners, and conduct business” [16]. ebXML describes itself as “a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet”, and allows “companies now have a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes”[17]. Further, ebXML positions itself (unsuccessfully to date) as an alternative to the Web services architectural components of UDDI and WSDL [16, 18]. However, there are aspects of ebXML that prove very useful to Web services and to this research with regard to business interaction standards. RosettaNet Partner Interface Processes® (PIPs®) also provide guidance for business interactions as well, but not necessarily with a Web service focus [19].

However, these approaches do not address the full life-cycle of activities to automate Web services. Additionally, these protocols do not address the aspects of agent specification and their operations, the design of knowledge repositories, and the high-level policies needed to guide the Virtual Organization.

Workflow is typically either centrally managed or virtually managed. The centrally managed model is the traditional enterprise model that has a single designated command and control for the business process. In the virtually managed model, the workflow operates in a VO where partners participate in a temporary relationship, and jointly “own” the process [20]. In this case, the workflow must act as a virtual command and control manager to ensure that the objectives of the process are properly executed. The workflow manager enforces Articles of Federation (AoF) [21] and Service Level Agreements (SLA) [22] established by the VO.

Articles of Federation (AoF) establish operating constraints and agreements for enacting workflows through the system. Specialized modules manage the terms before and during operations. For this research, AoF are the over-arching rules for participating in the federation, or VO. The AoF are managed through such mechanisms as Trust Pyramids and SLAs. This paper focuses on the policies aspect of AoF that governs the operations of Web services within the VO [21].

The management of Web services typically involves monitoring quality of service (QoS) performance properties against established SLAs. QoS is a combination of several qualities or properties of a service, such as *Availability*, *Security*, *Response time* and *Throughput*. Providers must manage and monitor the load they receive from requestors and check whether the service they provide to them meets the agreed-upon SLAs. Users must also check on the QoS they obtain. QoS monitoring may be outsourced to QoS monitoring services such as the ones that monitor Web sites (such as [www.keynote.com](http://www.keynote.com)) [23].

Although monitoring and analysis are considered important tasks of workflow management systems (WfMS) and business process management, little work has been done in developing a solution for integrating and analyzing the workflow audit trail information. Some approaches emphasize the need for integrating audit trail into data warehouse systems, while others are limited to a smaller set of workflow

history that is managed within a workflow management system. See McGregor [24] for a summary of related work in measuring business performance with Web Services.

Understanding the workload of Web and e-commerce sites is a fundamental step in sizing the IT infrastructure that supports these sites and in planning for their evolution so that QoS goals are met within cost constraints. One of the primary potential benefits of properly characterizing the workload is improve the quality of a requestor's experience with the service. Once one understands what customers do, what navigational patterns they follow, one can improve their experience and even attempt to dynamically identify a customer with a well-known behavior in order to provide a better experience for the customer, (i.e., personalized services, and maybe generate more profit to e-commerce sites) [25].

### **3. Knowledge-based Dynamic Semantic Web Services Framework**

The KDSWS Framework proposes to deliver a comprehensive and integrated end-to-end solution to dynamically deliver SWS. The framework positions itself as an enterprise-scale foundation to ultimately provide VOs with the interfaces necessary to interoperate via Semantic Web Services using, and quickly adapting to, the plethora of prevailing technologies and protocols. The importance of providing a comprehensive solution stems from incorporating issues at the front-end of the specification. By addressing isolated aspects of the problem space, current solutions are continually adding protocols to the 'protocol stack' [10] in order to cover new gaps previously not addressed. The importance of specifying an integrated methodology reduces the overhead of dealing with the nuances between disparate technologies. The addition of workflow management allows us to handle situations that require conditions such as: specific ordering of events (e.g., credit card needs to be approved before order is confirmed), parallel processing is necessary to reduce turnaround (e.g., inventory availability can be checked while the credit card is being approved), or conditional processing e.g., credit card is denied).

Like the Web Services Modeling Framework (WSMF) [26], ontologies and mediation are an integral part of this framework, and are manifested in the mapping and heuristics. The Web Services Modeling Ontology (WSMO) is an emerging initiative that is built on WSMF, and has a mission to "further standardization in the area of Semantic Web Service languages and to work toward a common architecture and platform for Semantic Web Services". A family of WSMO-centric technologies such as Web Services Modeling Language (WSML), Web Services Execution Environment (WSMX), and WSMO Editor are being targeted to provide a more comprehensive delivery environment for SWS [27]. However, it is yet to be determined how comprehensive and integrated this family of technologies and standards will remain.

This framework has similar guiding principles, with respect to incorporating knowledge and providing a 'protocol-independent' (vs. language-independent) conceptual base, to those found in [6]. However, this framework approaches the solution spaces from a different perspective by placing more emphasis on a total enterprise solution and by providing a 'way-forward' via a guiding methodology as to how to use the structures.

The structure and design of the components target, or map to, ontological implementations such as OWL and OWL-S, albeit with potential extensions to facilitate the additional functionality. OWL-S provides basic process and description capabilities, but it needs expansion and refinement to accommodate end-to-end functionality. A primary function of the heuristics is to enhance the mappings ability to dynamically mediate between disparate components. For more details about the KDSWS Framework, see Howard [3].

## **4. Delivering Web Services**

### **4.1 Delivery Context**

The delivery activities involve managing an execution of Web services via workflow specifications in order to deliver functionality and products to the requestor. The processes shown in Figure 3 are select

portions of the KDSWS Methodology's Deliver task. As a note, the items prefixed with "kdsd" denote KDL objects involved in the flow, and items prefixed with "kdsp" denote KDSPL objects used in the process.

The diagram delimits, by the vertical separator lines, the agents (e.g., Workflow and Fulfillment Agents) involved in the transformation process. The methodology shows the processes within the context of the KDSWS Threads. The flow of the KDL (data) objects is identified by the dashed line, where the execution of the KDSPL (process) objects is identified via the solid lines.

To distinguish between processes in the KDSWS methodology (a KDSPL-Process) and processes specified by instantiations of the methodology, the terms adjacent-process, macro-process, methodology-process, decision-process and simultaneous-process are used on items that pertaining to the methodology. Adjacent-processes are other processes in the KDSWS Methodology that are not detailed within this discussion. Macro-processes are the groupings of detailed methodology-processes in the diagram, where the decision-processes are the diamond-shaped processes that indicate a question to control the flow of the methodology. Simultaneous-processes are processes that are invoked by master processes to run with the master process to handle specialized aspects.

#### **4.2 Process vs. Workflow vs. Transaction**

The distinction between process, workflow, and transaction is somewhat blurred across the industry. Generally, a business process details the desired outcome and functional boundaries and is specified by Business Process Management (BPM) technologies. Workflow is considered to be the automation or full specification of resources for a process, while transaction control deals with maintaining the ACID (Atomic, Consistent, Isolation, and Durable) properties across distributed networks [28-30].

The fundamental elements across all three components (process, workflow and transactions) are a series of tasks or activities, resources that are responsible for each task, and an operator to indicate how the task is to stage the next steps to be executed. This research treats these elements to be steps, delegates and step-successor-modes; where steps are the tasks (e.g., `kdspIterateThroughSteps`), delegates are the resources (e.g., `kdsdWorkflowIterator`), and the operators are the step-successor-modes (e.g., sequential, parallel, do while). As one navigates from process to workflow to transaction, the level of detail and the scope of activity changes.

Additionally, this research addresses the issues related to configuring and coordinating multi-layered workflows that are encountered when a Web service that invokes a workflow subsequently calls other Web services that may also invoke workflows. Transactions are considered to be the finest grain of detail in a complex Web service, and are where the Web services are actually invoked. The transaction may be a single service or a group of services depending on the optimum execution scenario with regards to performance (see discussion below regarding the Enactment Package). The workflow manages the feedback as well as any anomalies (e.g., timeout, constraint violation) that may occur.

Some Web service-oriented processes/workflow solutions specify (identify) the Web services to be used as is the case with BPEL4WS. However, implementing the dynamic facets of this research can potentially yield better results by 1) having the request more fully profiled that will provide a closer match with the provider, and 2) taking advantage of higher quality (i.e., services are faster, cheaper and more accurately match the request) services.

The interoperation with specialty engines such as Workflow Management Systems (WfMS) or Expert Systems enhances the capabilities of this framework by providing mappings to these specialized tools. This mapping can be either be an import or interactive class, but the specialty data is generally imported to allow the specialty engine 'specialize' and be the master of its data.

### 4.3 *The Delivery Management Process*

The Manage Delivery KDSPL-process is preceded by the Broker Services and Deploy Services adjacent-processes, and is controlled by the Broker and Deploy agents. The Broker Services adjacent-process uses information from the WSDL+, UDDI+, and the Master Request object to provide Master Services, where the Master Request is the original request containing such information as constraints and preferences. The “+” notation on WSDL and UDDI reflects the semantically-enhanced versions of these protocols in order to facilitate the automation. The Master Services are the services that are dynamically matched, negotiated and configured to best fulfill the request. The services are potentially complex Web services; thus, they are referred as the “master” of to the Web services that they call. The Deploy Services adjacent-process applies the Articles of Federation (e.g., agreements are in tact), coordinates the resources (e.g., products are available), creates the Fulfillment Package in order to facilitate the managing and monitoring of the delivery, checks and opens ports in the case of critical endpoints, establishes time and security boundaries, profiles the environment, and captures the profile of the request.

The Fulfillment Agent, as the owner agent, controls the Invoke Master Request methodology-process that commences the execution steps by looking at the Fulfillment Package to potentially adjust resource allocations per the results of Deploy Services. (Designating the fulfillment agent as the owner may seem to be inconsistent with the diagram, but the Workflow Agent controls a looping control for the Fulfillment Agent.) If the adjustment is radical enough (determined by heuristics specified in the KDSPL), the process may be sent back to the Broker Services for “re-brokering”. This adjustment determination is done at the Invoke Master Request, versus the Deploy Services, because some long-standing arrangements may skip the Deploy Services stage.

The Manage Fulfillment methodology-process is a supervisory process over the simultaneous processes that handle the issues concerning the threads. The Articles of Federation, implemented as a set of rules for the policies of the VO and partially contained in the Fulfillment Package, are used to direct the activities at during these processes.

The Fulfillment Agent delegates work to the Workflow Agent in order for the Enact Workflow simultaneous-macro-process (simultaneous-macro-process are combined form of process types, with their sub-processes subsequently explained) to control the execution of the workflow. The Workflow? decision-process determines if multiple steps are necessary to be controlled by the Iterate through Steps methodology-process. This determination is made by either an explicit indicator or by interrogating the structure of service. These multiple-step services are sometimes referred to as Complex Web Services. The Manage Alternative Paths simultaneous-process runs along side of its master process, Iterate through Steps, to adjust the workflow as problems or more optimal alternatives are potentially encountered. The Configure Transactions methodology-process builds an Enactment Package, while the Execute Transactions methodology-process manages the execution of that Enactment Package. An Enactment Package is a group of services that can be executed together. In the case where two independent services from the same provider, the overhead of such activities as authenticating the transaction and making the network hop can be done once for the group versus multiple times for each service. The Secure Transaction adjacent-process handles the authentication, authorization and non-repudiation issues of the execution of the Enactment Package.

The Handle Anomalies simultaneous-macro-process uses Anomaly Policies to direct how to handle such situations as errors, exceptions and constraint violations. The Determine Severity methodology-process matches the anomalies against the prescribed conditions in Anomaly Policies, and then seeks available options in the Discover Alternatives methodology-process for the condition. The Adjust Process methodology-process chooses a method to handle the condition, and possibly adjust the workflow as well. However, not all alternatives will realize an adjustment to processes; some may simply abort or change an option.

The Manage SLA simultaneous-macro-process manages the Quality of Service aspects by enforcing the Service Level Agreements. The Provision Resources methodology-process allocates the resources to support the request, while the Manage Workload methodology-process balances the load on those resources to the agreed-upon levels. The Monitor Compliance methodology-process ensures that providers and requestors abide by the SLA terms. Violations are passed off to the Handle Anomalies simultaneous-macro-process.

The Integrate w/ other services and protocols simultaneous-macro-process coordinates mediates the interactions with the surrounding and disparate interfaces by using Mediation Policies in the Mediate Semantics methodology-process. The Establish Interfaces methodology-process identifies which interfaces require mediation.

The Manage Payloads simultaneous-macro-process manages the messaging activity. The Determine Message Structure methodology-process decides such items as which protocol is most appropriate to use per the profile characterized by the Profile Payload methodology-process and guidelines prescribed in Payload Policies. The Monitor Payload methodology-process ensures that the messages are being adequately processed, with problems being passed off to the Handle Anomalies simultaneous-macro-process.

The Monitor & Measure Fulfillment simultaneous-macro-process supports the metrics needs of the delivery processes, and helps ensure that the interactions with the requestor, albeit human or machine, are effectively being handled. The bi-directional feedback spoken of earlier is kept in the Resource History for the VO's resources' (e.g., providers, requestors, agents) activity per the role they take on in the process, while the Audit Logs, also spoken of earlier, are used with Resource History in this macro-process. The Performance Metrics are maintained by the Monitor Workflow methodology-process and the Track Resource Performance methodology-process, and are gauged against the Metric Goals and Rating Structures in the Measure Appropriateness & Fit methodology-process for potential corrective action as needed.

The Inventory, Delivery Resources (including polices) and the Fulfillment Package (that conveys Master Request preferences) feed into the Distribute Products macro-process in order to determine whether products are delivered, notifications sent, or responses sent to requestors immediately as the individual services are executed or packaged and held until the entire workflow is complete. The Manufacture Deliverable methodology-process produces the desired product in such forms as information provided (e.g., status of the request) or functionality performed (e.g., an order is placed). This manufacturing may take extended periods of time, which may factor in the feedback timing and form. The Manage Inventory simultaneous-process maintains acceptable inventory levels per enterprise and local Constraints. The Establish Route methodology-process determines the appropriate and delivery vehicle (e.g., email, http) for the product.

The Deliver functionality & provide feedback simultaneous-macro-process handles the actual delivery of the product and communicates with the requestor. The Respond to Master Request methodology-process sends notifications to the requestor in form of either a Results Template or Status Template. The Collect Directions methodology-process prompts for feedback for additional instructions from the requestor, albeit from a human or application. This human intervention is especially useful to specify behavior when automation cannot effectively make judgment decisions in the case of such events as anomalies. The Deliver Product methodology-process sends the deliverable to the requestor.

To illustrate the KDL, Figure 1 shows a partial specification of the Master Request Profile. The constraint on kdsdSearchDomain specifies another object, kdsdValidSearchDomain, to reflect that the field should contain a valid value. The TYPEs with a value of :object (e.g., kdsdRequest, kdsdRequestProfile, kdsdSearchPriorities, etc.) reflect that those attributes are aggregated into this

object. The object-wide CONSTRAINT (Do not use....) is indicated as text here (and in the discussion below as well) for easier presentation, but would normally be specified in a formal constraint language. The HUERISTIC instructs agents to not use the profile if it is stale (that is, it has not been used in over six months). The METHODS listed denotes that agents have the Adjust Master Request Profile process available for use on this object.

object-type::=kdsdMasterRequest				
:SUPERTYPES	kdsdProfile			
:ATTRIBUTES	kdsdSearchDomain	:TYPE	string	:CONSTRAINT kdsdValidSearchDomain
	kdsdVersionNumber	:TYPE	number	:CONSTRAINT
	kdsdVersionDate	:TYPE	date_time	
	kdsdLastUsedDate	:TYPE	date_time	:CONSTRAINT Cannot be in future
	kdsdRequest	:TYPE	object	
	kdsdRequestorProfile	:TYPE	object	
	kdsdSearchPriorities	:TYPE	object	
	kdsdRequestConstraints	:TYPE	object	
	kdsdRequestPreferences	:TYPE	object	
	kdsdGoals	:TYPE	object	
:CONSTRAINT	Do not use if kdsdRequestorProfile and kdsdRequest are not specified			
:HEURISTICS	if kdsdLastUsedDate > 6 months old, do not use			
:METHODS	kdspAdjustMasterRequestProfile			

**Figure 1. kdsdMasterRequestProfile KDL Specification**

Figure 2 shows a partial KDSPL specification for the Enact Workflow process. The GOALS are embodied in the kdsdEnactWorkflowGoal object. The agent can make decisions unique to the context via the TASK being set to kdspDeliver and THREAD being set to kdspManagement. The OWNER is the kdsdFulfillmentAgent, and the STEWARD is the kdsdWorkflowAgent. The PREDECESSORS of this macro-process is kdspInvokeMasterRequest, and the next step (the SUCCESSOR) in line is kdspSecureTransaction.

To help readability, STEPNAME's are highlighted and some of the discussion items are highlighted in gray for the STEP discussion. The STEP-SUCCESSOR-BRANCH designates kdspIterateThroughSteps when WorkflowSpecified="Yes" because of its specification in STEP-CONTROL-CONDITION. However, notice that an additional STEP-SUCCESSOR-BRANCH exists to further specify that the step kdspIterateThroughSteps is iterated through until EndOfSteps="Yes".

The SEQUENCE-NUMBERS branch off (i.e., 1.1 for kdspIterateThroughSteps and 1.2 for kdspSecureTransaction as sub-processes of kdspEnactWorkflow) for the downstream processes. The DELEGATE'd agent (specified by the DELEGATE-TYPE is "Agent") for kdspIterateThroughSteps is kdsdWorkflowIterator in a "Support" DELEGATE-ROLE. The STEP-SUCCESSOR-MODE of "Simultaneous" indicates that kdspIterateThroughSteps is the master process for the kdspManageAlternatePaths.

The STEP-CONTROL-CONDITION of "Critical Error" exemplifies the ability to specify human intervention. The associated STEP-SUCCESSOR-BRANCH of kdspCriticalError delegates the work to another process, kdspNotifyError. The DELEGATE-TYPE of "human" designates that a human needs to handle this step, and A DELEGATE-ROLE of "Alternative" designates that the step is not in the typical path. The OPERATION parameter in the kdspCriticalError step specifies the class errorHandler's method notifyError, which will prompt a decision from a list of prescribed actions (not shown). The STEP-CONTROL-CONDITION of "Hueristic #2" demonstrates the dynamic adjustment ability of the framework to alter the path of execution based on rules. The SEMANTICWEBMAP maps the DELEGATE for the kdspIterateThroughSteps object to an OWL-format specification called iterateWorkflowPattern.

object-type::=kdspEnactWorkflow																																																																																	
:GOALS	kdspEnactWorkflowGoal (Manage the enactment of a workflow)																																																																																
:TASK	kdspDeliver																																																																																
:THREAD	kdspManagement																																																																																
:OWNER	kdspFulfillmentAgent																																																																																
:STEWARD	kdspWorkflowAgent																																																																																
:PREDECESSORS	kdspInvokeMasterRequest																																																																																
:SUCCESSORS	kdspSecureTransaction																																																																																
:STEPS	<table border="1"> <tr> <td>:STEPNAME</td> <td>kdspTestWorkflowSpecified</td> </tr> <tr> <td>:SEQUENCE-NUMBER</td> <td>1</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Decision</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Iterator</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspIterateThroughSteps</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Sequential</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspSecureTransaction</td> </tr> <tr> <td>:STEP-CONTROL-CONDITION</td> <td>EndOfSteps="Yes"</td> </tr> <tr> <td>:STEP-CONTROL-CONDITION</td> <td>WorkflowSpecified="Yes"</td> </tr> <tr> <td>:STEPNAME</td> <td>kdspIterateThroughSteps</td> </tr> <tr> <td>:SEQUENCE-NUMBER</td> <td>1.1</td> </tr> <tr> <td>:DELEGATE</td> <td>kdspWorkflowIterator</td> </tr> <tr> <td>:DELEGATE-TYPE</td> <td>Agent</td> </tr> <tr> <td>:DELEGATE-ROLE</td> <td>Support</td> </tr> <tr> <td>:OPERATION</td> <td>iterateWorkflowSteps</td> </tr> <tr> <td>:METHOD-NAME</td> <td>kdspWorkflowIterator.Iterate</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Simultaneous</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspManageAlternatePaths</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Sequential</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspConfigureTransactions</td> </tr> <tr> <td>:STEPNAME</td> <td>kdspConfigureTransactions</td> </tr> <tr> <td>:SEQUENCE-NUMBER</td> <td>1.2</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Sequential</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspSecureTransaction</td> </tr> <tr> <td>:STEPNAME</td> <td>kdspManageAlternatePaths</td> </tr> <tr> <td>:SEQUENCE-NUMBER</td> <td>1.1.1</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Decision</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspCriticalError</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Decision</td> </tr> <tr> <td>:STEP-SUCCESSOR-BRANCH</td> <td>kdspRequisitionBlankets</td> </tr> <tr> <td>:STEP-CONTROL-CONDITION</td> <td>"Critical Error"</td> </tr> <tr> <td>:STEP-CONTROL-CONDITION</td> <td>Huristic #2</td> </tr> <tr> <td>:STEPNAME</td> <td>kdspCriticalError</td> </tr> <tr> <td>:SEQUENCE-NUMBER</td> <td>1.1.1.1</td> </tr> <tr> <td>:STEP-SUCCESSOR-MODE</td> <td>Sequential</td> </tr> <tr> <td>:DELEGATE</td> <td>kdspNotifyError</td> </tr> <tr> <td>:DELEGATE-TYPE</td> <td>Human</td> </tr> <tr> <td>:DELEGATE-ROLE</td> <td>Alternative</td> </tr> <tr> <td>:OPERATION</td> <td>errorHandler.notifyError</td> </tr> <tr> <td>:MANUAL INTERVENTION</td> <td>Ask human to make a decision for next action</td> </tr> </table>	:STEPNAME	kdspTestWorkflowSpecified	:SEQUENCE-NUMBER	1	:STEP-SUCCESSOR-MODE	Decision	:STEP-SUCCESSOR-MODE	Iterator	:STEP-SUCCESSOR-BRANCH	kdspIterateThroughSteps	:STEP-SUCCESSOR-MODE	Sequential	:STEP-SUCCESSOR-BRANCH	kdspSecureTransaction	:STEP-CONTROL-CONDITION	EndOfSteps="Yes"	:STEP-CONTROL-CONDITION	WorkflowSpecified="Yes"	:STEPNAME	kdspIterateThroughSteps	:SEQUENCE-NUMBER	1.1	:DELEGATE	kdspWorkflowIterator	:DELEGATE-TYPE	Agent	:DELEGATE-ROLE	Support	:OPERATION	iterateWorkflowSteps	:METHOD-NAME	kdspWorkflowIterator.Iterate	:STEP-SUCCESSOR-MODE	Simultaneous	:STEP-SUCCESSOR-BRANCH	kdspManageAlternatePaths	:STEP-SUCCESSOR-MODE	Sequential	:STEP-SUCCESSOR-BRANCH	kdspConfigureTransactions	:STEPNAME	kdspConfigureTransactions	:SEQUENCE-NUMBER	1.2	:STEP-SUCCESSOR-MODE	Sequential	:STEP-SUCCESSOR-BRANCH	kdspSecureTransaction	:STEPNAME	kdspManageAlternatePaths	:SEQUENCE-NUMBER	1.1.1	:STEP-SUCCESSOR-MODE	Decision	:STEP-SUCCESSOR-BRANCH	kdspCriticalError	:STEP-SUCCESSOR-MODE	Decision	:STEP-SUCCESSOR-BRANCH	kdspRequisitionBlankets	:STEP-CONTROL-CONDITION	"Critical Error"	:STEP-CONTROL-CONDITION	Huristic #2	:STEPNAME	kdspCriticalError	:SEQUENCE-NUMBER	1.1.1.1	:STEP-SUCCESSOR-MODE	Sequential	:DELEGATE	kdspNotifyError	:DELEGATE-TYPE	Human	:DELEGATE-ROLE	Alternative	:OPERATION	errorHandler.notifyError	:MANUAL INTERVENTION	Ask human to make a decision for next action
:STEPNAME	kdspTestWorkflowSpecified																																																																																
:SEQUENCE-NUMBER	1																																																																																
:STEP-SUCCESSOR-MODE	Decision																																																																																
:STEP-SUCCESSOR-MODE	Iterator																																																																																
:STEP-SUCCESSOR-BRANCH	kdspIterateThroughSteps																																																																																
:STEP-SUCCESSOR-MODE	Sequential																																																																																
:STEP-SUCCESSOR-BRANCH	kdspSecureTransaction																																																																																
:STEP-CONTROL-CONDITION	EndOfSteps="Yes"																																																																																
:STEP-CONTROL-CONDITION	WorkflowSpecified="Yes"																																																																																
:STEPNAME	kdspIterateThroughSteps																																																																																
:SEQUENCE-NUMBER	1.1																																																																																
:DELEGATE	kdspWorkflowIterator																																																																																
:DELEGATE-TYPE	Agent																																																																																
:DELEGATE-ROLE	Support																																																																																
:OPERATION	iterateWorkflowSteps																																																																																
:METHOD-NAME	kdspWorkflowIterator.Iterate																																																																																
:STEP-SUCCESSOR-MODE	Simultaneous																																																																																
:STEP-SUCCESSOR-BRANCH	kdspManageAlternatePaths																																																																																
:STEP-SUCCESSOR-MODE	Sequential																																																																																
:STEP-SUCCESSOR-BRANCH	kdspConfigureTransactions																																																																																
:STEPNAME	kdspConfigureTransactions																																																																																
:SEQUENCE-NUMBER	1.2																																																																																
:STEP-SUCCESSOR-MODE	Sequential																																																																																
:STEP-SUCCESSOR-BRANCH	kdspSecureTransaction																																																																																
:STEPNAME	kdspManageAlternatePaths																																																																																
:SEQUENCE-NUMBER	1.1.1																																																																																
:STEP-SUCCESSOR-MODE	Decision																																																																																
:STEP-SUCCESSOR-BRANCH	kdspCriticalError																																																																																
:STEP-SUCCESSOR-MODE	Decision																																																																																
:STEP-SUCCESSOR-BRANCH	kdspRequisitionBlankets																																																																																
:STEP-CONTROL-CONDITION	"Critical Error"																																																																																
:STEP-CONTROL-CONDITION	Huristic #2																																																																																
:STEPNAME	kdspCriticalError																																																																																
:SEQUENCE-NUMBER	1.1.1.1																																																																																
:STEP-SUCCESSOR-MODE	Sequential																																																																																
:DELEGATE	kdspNotifyError																																																																																
:DELEGATE-TYPE	Human																																																																																
:DELEGATE-ROLE	Alternative																																																																																
:OPERATION	errorHandler.notifyError																																																																																
:MANUAL INTERVENTION	Ask human to make a decision for next action																																																																																
:CONSTRAINT	kdspRetryLimitOf3																																																																																
:HEURISTICS	#1:Rank the alternative paths according to priorities provided by the ranking profile kdspRankAlternativePath #2:If kdspLocalBlanket ConstraintViolation, Execute kdspLocalBlanketConstraintViolation																																																																																
:MAP	<table border="1"> <tr> <td>:FUNCTION-NAME</td> <td>kdspIterateThroughSteps.DELEGATE</td> </tr> <tr> <td>:PROTOCOL</td> <td>OWL</td> </tr> <tr> <td>:OBJECT-NAME</td> <td>iterateWorkflowPattern</td> </tr> </table>	:FUNCTION-NAME	kdspIterateThroughSteps.DELEGATE	:PROTOCOL	OWL	:OBJECT-NAME	iterateWorkflowPattern																																																																										
:FUNCTION-NAME	kdspIterateThroughSteps.DELEGATE																																																																																
:PROTOCOL	OWL																																																																																
:OBJECT-NAME	iterateWorkflowPattern																																																																																

Figure 2. kdspEnactWorkflow KDSPL Specification

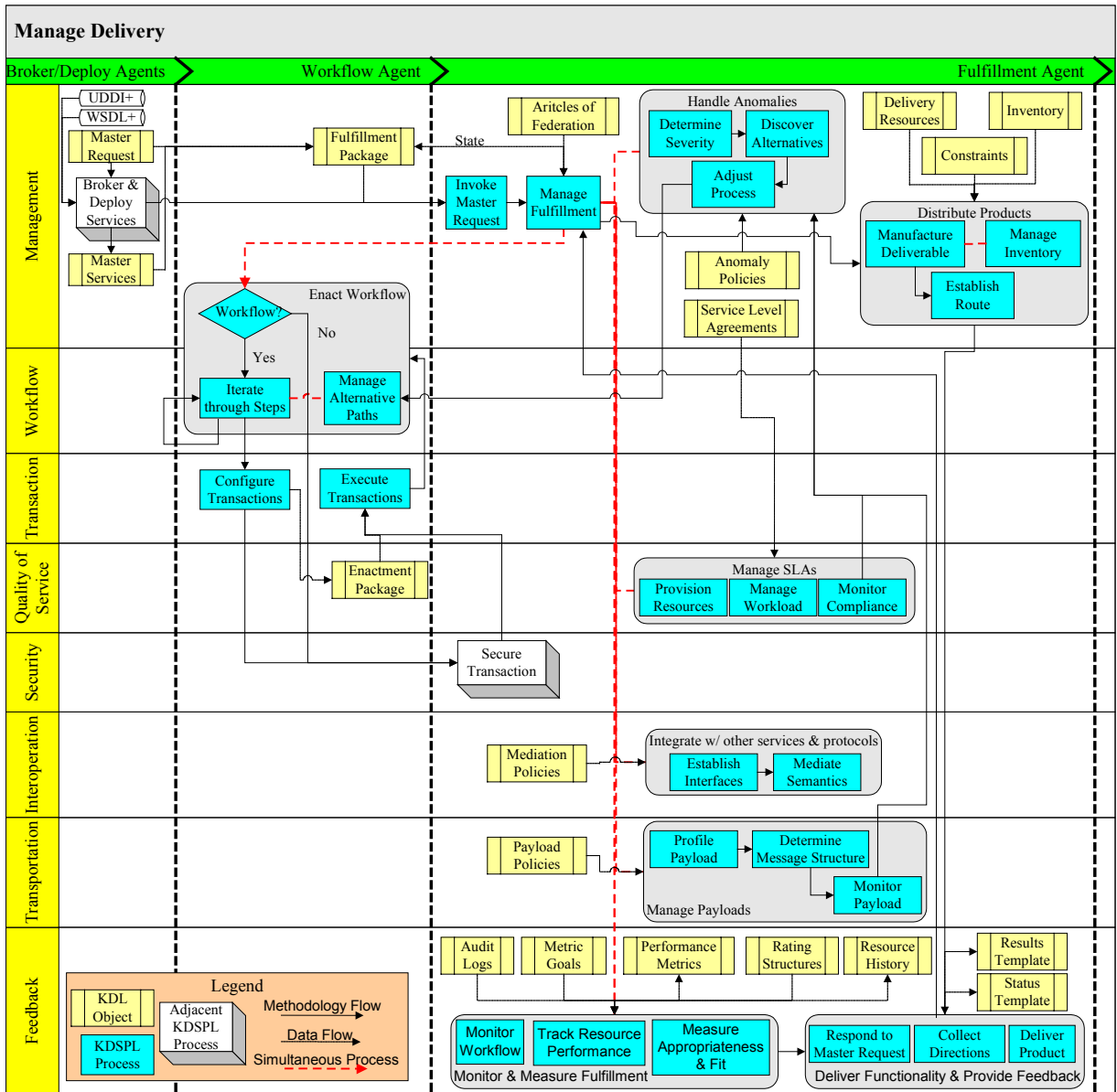


Figure 3. Managing the Delivery of Semantic Web Services

## 5. Conclusions

In order to accelerate the introduction of new concepts and systems, we need approaches which automate the entire Web services life-cycle and address the issues related to the creation, management and execution of services in Virtual Organizations. This paper presents an agent-based approach to managing the delivery of Semantic Web Services for use within a Virtual Organization, and is part of a larger methodology, called the KDSWS Framework. The framework provides a formal model-based approach to implementing Web services that specifies the modeling, specification, design, implementation and deployment of systems composed of SWS. The goal of the framework is to support the automatic discovery, composition, execution and management of Web services for the Virtual Organization in a protocol-independent manner.

The key contributions to the delivery of Semantic Web Services discussed in this paper are: a managed approach to fulfill requests, a workflow-based methodology to control processes and adapt to unexpected

conditions, and a set of feedback processes that facilitate both delivery and optimization methods. The issues that face Virtual Organizations were also addressed, along with a means to specify human interaction to deal with anomalies that occur in the process.

Finally, this managed approach for the delivery of Semantic Web Services within a knowledge-based framework provides the ability to dynamically specify, create, and execute Web services for a Virtual Organization's workflow needs. The enterprise issues are separated, thus providing a crisper and tighter design in order to crystallize the concepts that are specified. This ultimately provides a clearer set of roles and responsibilities of the architectural components within the system. The comprehensive treatment of issues concerning the delivery of services reduces the vulnerability of gaps in functionality. The integrated specification is more adaptable due to providing a consistent foundation from which to deal with the disparate technologies. The ability to dynamically specify and alter behavior delivers much more responsive applications.

## Acknowledgements

This work was sponsored by a NURI from the National Geospatial-Intelligence Agency (NGA). This work was also supported in part by the Advanced Research and Development Activity (ARDA). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U. S. Government.

## 6. References

- [1] D. Hollingsworth, "Workflow Management Coalition. Workflow Reference Model," 1995.
- [2] D. C. Marinescu, *Internet-based Workflow Management: Toward a Semantic Web*. New York: Wiley-Interscience, 2002.
- [3] R. Howard and L. Kerschberg, "A Framework for Dynamic Semantic Web Services Management," *Special Issue on Service Oriented Modeling, International Journal in Cooperative Information Systems, Accepted for Publication*, 2004.
- [4] R. Howard and L. Kerschberg, "A Knowledge-based Framework for Dynamic Semantic Web Services Brokering and Management," presented at Database and Expert Systems Applications (DEXA) 2004, WebS Workshop, Zaragoza, Spain, 2004.
- [5] M. Paolucci, K. Sycara, and T. Kawamura, "Delivering Semantic Web Services," 2003.
- [6] A. Gomez-Perez, R. Gonzalez-Cabero, and M. Lama, "A Framework for Design and Composition of Semantic Web Services," presented at 2004 AAAI Spring Symposium Series, Stanford University Palo Alto, CA, 2004.
- [7] P. Mika, M. Sabou, A. Gangemi, and D. Oberle, "Foundations for OWL-S: Aligning OWL-S to DOLCE," presented at 2004 AAAI Spring Symposium Series, Stanford University Palo Alto, CA, 2004.
- [8] M. Paolucci, J. Soudry, N. Srinivasan, and K. Sycara, "A Broker for OWL-S Web services," presented at 2004 AAAI Spring Symposium Series, Stanford University Palo Alto, CA, 2004.
- [9] DAML.org, "OWL-S: Semantic Markup for Web Services," 2003.
- [10] L. Wilkes, "The Web Services Protocol Stack," CBDI Web Services Roadmap, 2004.
- [11] D. Briukhov, L. Kalinichenko, and I. Tyurin, "Extension of Compositional Information Systems Development for the Web Services Platform," in *Advances in Databases and Information Systems*, vol. 2798/2003, 2003, pp. 16-29.
- [12] A.-W. a. N. Scheer, Markus, *ARIS Architecture and Reference Models for Business Process Management*, 2000.
- [13] IBM, "Specification: Business Process Execution Language for Web Services Version 1.1," 2003.
- [14] A. Arkin, "Business Process Modeling Language," BMPI.org, 2002.
- [15] W3C, "Web Services Choreography Description Language 1.0," 2004.
- [16] B. Hofreiter, C. Huemer, and W. Klas, "eXML: status, research issues, and obstacles," presented at Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002. Proceedings. Twelfth International Workshop on, 2002.

- [17] ebXML, "About ebXML," 2004.
- [18] S. Patil and E. Newcomer, "ebXML and Web services," *Internet Computing, IEEE*, vol. 7, pp. 74-82, 2003.
- [19] RosettaNet, "RosettaNet PIPs," 2004.
- [20] R. L. Reid, K. J. Rogers, M. E. Johnson, and D. H. Liles, "Engineering the Virtual Enterprise," Automation & Robotics Research Institute.
- [21] L. Kerschberg, H. Gomma, D. Menasce, and J. P. Yoon, "Data and Information Architectures for Large-Scale Distributed Data Intensive Information Systems," *Proceedings Eighth International Conference on Statistical and Scientific Database Management*, 1996.
- [22] A. Dan, H. Ludwig, and G. Pacifici, "Web service differentiation with service level agreements," 2003.
- [23] D. A. Menasce, "QoS issues in Web services," *IEEE Internet Computing*, vol. 6, pp. 72-75, 2002.
- [24] C. McGregor and J. Scheifer, "A framework for analyzing and measuring business performance with Web services," presented at IEEE International Conference on E-Commerce (CEC 2003), 2003.
- [25] D. Menascé, B. Abrahao, D. Barbara, V. Almeida, and F. Ribeiro, "Fractal Characterization of Web Workloads," presented at The Eleventh International World Wide Web Conference (WWW 2002), Web Engineering Track, Honolulu, Hawaii, 2002.
- [26] D. Fensel and C. Bussler, "The Web Service Modeling Framework WSMF."
- [27] WSMO, "Web Services Modeling Ontology," vol. 2004, 2004.
- [28] M. Virdell, "Business processes and workflow in the Web services world," 2003.
- [29] M. P. Papazoglou, "Web Services and Business Transactions," 2003.
- [30] I. Ghalimi, "BPM vs. Workflow," 2003.