Singapore Management University
Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

7-2005

Middleware Architecture for Evaluation and Selection of 3rd Party Web Services for Service Providers

Dipanjan CHAKRABORTY
IBM India Research Laboratory

Suraj K. JAISWAL
IBM India Research Laboratory

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Amit A. NANAVATI
IBM India Research Laboratory

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

🎨 Part of the Software Engineering Commons

Citation

Middleware Architecture for Evaluation and Selection of 3rd Party Web Services for Service Providers

Dipanjan Chakraborty, Suraj Kumar Jaiswal, Archan Misra, Amit A. Nanavati

Abstract— This paper presents an architecture to facilitate efficient evaluation and selection of 3rd party web services for service providers. Most service provider architectures have primarily focused on providing web service front ends to legacy systems, aggregating and delivering services via workflows. These architectures primarily considered static business contracts between the service provider and its (web-service enabled) business partners. This approach makes these architectures inflexible to variations in business requirement, partners' performance and customer requirements. Our architecture provides a flexible means for service providers to optimize business performance. Based on the historical performance, extant context, and optimising business rules, the appropriate service is selected and invoked to serve a customer request. We have developed a prototype of our system, dubbed ODSS-in-ENDS that integrates business partners and displays the dynamic evaluation and selection of services.

I. INTRODUCTION

Service providers (**SP**s) primarily act as "service portals", offering a variety of services either using in-house solutions or through partnership with various 3rd party business services. In many instances, these business services act as content providers (**CP**) [1]. To enable a particular service, a SP might interact with only a single content provider or might leverage a *composite* workflow (using workflow technologies such as [1], [7]) involving the invocation of multiple 3rd party CPs. Examples of such interaction between SPs and CPs is presently observed in both the Internet Service Providers/Portal (such as Amazon.com or Earthlink) and the Telecom/Cellular Service Provider(e.g., Verizon, Vodafone, NTT DoCoMo) markets.

At present, SPs primarily depend on static business contracts with their CPs to enable service delivery. In many of the more sucessful Telecom Service Provider offerings (e.g., NTT Docomo), third-party services are tightly integrated into an SP-proprietary platform, based on custom APIs. Such a "walled-

Author names are in alphabetical order. Archan Misra works for IBM TJ Watson Research Center, Hawthorne, NY and the rest of them work at IBM India Research Lab, New Delhi, India. {cdipanjan,surajkj,namit}@in.ibm.com,archan@us.ibm.com. Corresponding author: Amit A. Nanavati,+91-11-5129-2216.

[1]We use the terms *content provider* and *business partners* interchangeably in this paper to refer to any external business partner, not necessarily restricted to content providers, which the SP leverages to offer its own service.

garden" IT model is, however, restrictive as it does not enable either the SP or the individual CPs to leverage the benefits of a *competitive, market-driven* service infrastructure. This problem will become specially acute with the adoption of universal service interface standards (such as Web Services [1], [3], [4]), which enable the invocation of services using a standard,language and platform-independent format that encapsulates all implementation-specific details. From the SP's viewpoint, the use of static contractual agreements prevents it from being able to dynamically select among competing content providers (CPs) (e.g., maps, florist, ticket reservations), based on changes in customer profile or context or on the performance profile of individual CPs. From the individual CP's viewpoint, the adoption of standard service invocation mechanisms allows it to associate and peer with multiple, usually competing, SPs, and thereby expand the CP's reach without an unacceptable increase in it's service implementation and management cost.

In this new vision of a market-based services environment, a SP would act as the middleware platform and mediate functions such communication, billing etc., offering value-added serices through the appropriate composition of external CP services. To enable such a vision, it is however imperative that the SP is able to *monitor, evaluate, rank and select* appropriate CPs and adapt to changing conditions. This is critical to the SP's business survival, since it acts as the interface to the individual customers, and is thus responsible for satisfying their quality expectations. While there is a great deal of ongoing work on mechanisms for dynamic service discovery and ontology-based composition, relatively little research has been done on mechanisms and processes by which an SP can a) monitor the performance of such services and b) base the dynamic selection of CP services on such monitoring and higher (business) level policies. In this paper, we address these problems by presenting a Web services-oriented service provider middleware architecture and implementation that integrates the performance monitoring of individual CPs, along with other dynamic contextual conditions, in the automatic selection of appropriate CPs. In our archi-

tecture, a CP's performance is evaluated not just in terms of network or service-level parameters, but also via other business processes (e.g., complaint handling). Moreover, the selection of multiple Web services occurs through a two-step process, including an initial filtering of a set of feasible workflows for each task depending on business agreements and end user needs, and a subsequent dynamic context-based selection of the most appropriate workflow. The basic operation of our system is captured in the following scenario:

SPan, a popular service provider, has relationships with many content providers (business partners), whom he uses to create and deliver value-added services to his customers. His partner providers have deployed web services which can be accessed by SPan; to deliver innovative services, SPan composes workflows that orchestrate various partner web services. SPan's relationship with multiple competing CPs is designed to safeguard himself from outages or other variation in the quality of his partner's services, and ensure 24x7 customer satisfaction to his customers. As with any business, SPan has both quality-sensitive and price-sensitive customers. Since SPan has different contracts (different pricing, minimum usage guarantees) with different CPs, SPan would like a way to select the optimal worflow for each customer, i.e., one that satisfies the possibly differentiated quality assurances provided to individual customers while optimizing (e.g., minimizing cost) SPan's use of partner resources. To stay competive and maintain customer satisfaction, SPan should be able to continually monitor the performance of his partners (at various time scales). Further, he should be able to select the "best" compositive workflow dynamically, taking into account contextual attributes such as the customer's location (a key attribute in many location-based services for cellular service providers) or the partner response times.

To our knowledge, our proposed architecture and implementation is novel in that it combines several different aspects of commercial SP operation, that are typically addressed in isolation. First, our framework is extensible and allows the performance of CP services to be monitored at a variety of time scales, both to ensure conformance to contractual QoS metrics and to additionally rank or rate them based on other non-contractual performance metrics. This allows a SP to not only meet quantifiable performance guarantees, but also maximize implicit customer satisfaction. Second, the static selection of alternative worflows for a specific task is also driven by SP *metarules* which often capture the business-level preferences of the SP and map them into the actual deployed IT service infrastructure. As a result of this, the service/workflow selection can be easily modified by chang-

ing the SP metarules. Finally, the actual service selection logic includes both a static (off-line) optimization component, and a dynamic self-managed context-sensitive selection mechanism. The static or off-line portion of the middleware uses (in addition to metarules) sets of performance logs and analyzers to capture performance histories of CPs and a set of alternative candidate workflows per task (service) as input. The output from this component is a subset of "feasible" workflows per task, along with a set of rules enumerating the predicates (some of which refer to attributes that change dynamically) under which a particular workflow should be executed. The run-time component of the system then uses an external context aggregator to retrieve these contextual attributes, and then selects the appropriate workflow from the feasible set based on the attribute values. This allows the SP to easily handle contingencies, such as service unavailability and unexpected traffic.

II. Architecture Overview

Figure 1 shows the logical architecture of our middleware. Two key aspects of our system are the (1) two-step context-cum-policy driven evaluation and selection of services, and (2) feedback-based control. Accordingly, the architectural components can be logically separated into two modules: *offline* and *online*. The offline module primarily carries out the offline *monitoring, evaluation and filtering* of workflows by considering slow-changing context (mostly associated with business context and rules between SP and CPs). The online module carries out the *selection* process by considering dynamic run-time context (mostly associated with the end user/client). The Context Server [8] (integrated in the online module) collects several run-time context data from both CPs and end users.

Users are often classified as belonging to several categories (e.g. business class, economy). We employ user profiles that capture such information and utilize it to personalize the content, apply preferences and privacy considerations while evaluating and selecting a service. This is used to generate the *metarules* that specify evaluation and selection criteria for the competing CPs. For example, a customer may be categorized as a price-sensitive customer depending on the type of service he has used before. So, a corresponding metarule will try to filter costly CPs while choosing a service (or a workflow) for the user.

Business context refers to service-level agreements such as minimum transactions required to obtain certain discounts, transaction volume guarantees, service failure contingencies etc. A set of *information logs and analyzers* monitor and track the performance of content provider services. The analyzers may be used to monitor statistical, aggregated, customer-
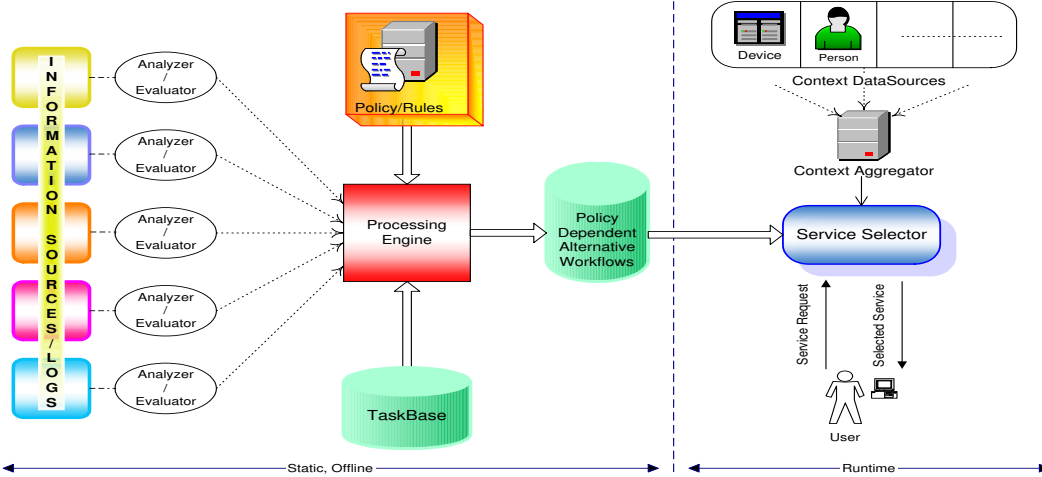
Fig. 1. Logical Architecture of Dynamic Service Evaluation and Selection Platform

centric, service-centric and/or location-dependent performance. The architecture provides for new analyzers to be plugged-in easily. Information analyzers (through the information logs) provide a feedback control to the system. This feedback is utilized to further refine the filtering, evaluation and selection mechanism.

Apart from user profiles, metarules are derived after considering various factors such as partner agreements, service invocation costs, customer segments, quality of service guarantees made to the customers. For example, a SP may decide to offer cheaper services to price-sensitive customers, and content-rich, quick response services to quality-sensitive customers. Metarules provide a mechanism for the SP to optimize cost/quality depending on his individual customers' needs and profiles.

A set of *alternative workflows* per task is provided through a task database. These workflows may be composed of alternative web services from various CPs, and may also include multiple instances of the same web service from a particular CP. The alternative workflows may be orchestrated using commonly used standards like BPEL [7]. Moreover, alternate workflows for the same task may be obtained from high-level planners [5].

The Processing Engine, forms the central processing unit of the architecture. The goal of this engine is to select and rank the alternative workflows based on the business optimization criteria provided in the business policies. It outputs a set of *context-dependent alternative workflows* which summarize the conditions under which a workflow should be executed.

The runtime component of the architecture consists of a *context aggregator* and a *service selector*. The context aggregator gathers context information

from various sources. The service selector obtains the context from the context aggregator, matches it with that in the set of context-dependent alternative workflows database to select the appropriate workflow/individual web service, and proceeds to execute it.

III. ODSS-IN-ENDS SYSTEM

The primary components our architecture dubbed ODSS-in-ENDS (**O**n **D**emand **S**ervice **S**election in **End**-to-End **S**ystems) are the *offline* and the *online* modules. The *offline* module further consists of the WorFare engine (the processing engine), the Partner Relationship Management (PRM) component, the Log Mining component and the various stores to keep the metarules, the set of exhaustive workflows for each service and the context-dependent alternative workflows. The *online* module primarily contains the runtime service selection platform and the Context Aggregator that collects several run-time information from content provider services. We further illustrate the metarule, business partner evaluation and other components of our architecture in the following subsections. In our descriptions and our prototype implementation, we shall use telecom service providers as a running example of a service provider.

A. Metarules

Each metarule consists of a *condition* and an *action*. A *condition* is formed by the context of the end user and consists of a conjunction of several parameters representing the context of the user. An *action* specifies *directives* to determine appropriate set of workflows for selection of a service. An *action* also enumerates *run-time context checks* (discussed in section III-E) that are to be executed before actually executing a particular service. Each *action* may con-

```
<Policy Name="PremiumAndPeakAndRemoteUsers">
<Condition>
    <UserCategory>premium</UserCategory>
    <Location>remote</Location>
    <NetworkStatus>peak</NetworkStatus>
</Condition>
<Actions>
  <HistoryFilters>
    <UserCategory />
    <Location />
    <NetworkStatus>peak</NetworkStatus>
    <Response>fast</Response>
  </HistoryFilters>
  <PRMFilters>
    <PRM-Ranking>
      <Operator>less</Operator>
      <Value>4</Value>
    </PRM-Ranking>
  </PRMFilters>
  <PriceFilters>
    <Operator>leq</Operator>
    <Value>0.75</Value>
    <ValueCurrency></ValueCurrency>
  </PriceFilters>

  <ContextChecks>
    <Response>Fast</Response>
  </ContextChecks>
 </Actions>
</Policy>
```

Fig. 2. XML-ised representation of a *metarule* in ODSS-in-ENDS

sist of several *directives*. Each *directive* specifies evaluation criteria from various service evaluation components. This is used by the WorFare engine to select and rank possible alternative workflows to deliver a particular service.

A metarule may incorporate any service-level agreement [13] between the service provider and the content providers. Rules specify several constraint checks that enable better selection of the appropriate workflow. The service provider may write metarules that are geared towards optimizing quality-of-service parameters (like service execution time) while maintaining service-level agreements and ensuring partner satisfaction. Figure 2 shows the XML-ised representation of a metarule in our system.

B. Evaluation of Business Partner Services

Business Partner services are evaluated in several different ways by several pluggable modules in ODSS-in-ENDS. Services are evaluated with respect to the existing business contracts existing between the partner and the service provider. Services are also evaluated with respect to absolute performance parameters such as average execution time that may not be part of a business contract. Evaluation is carried out based on long-term history, as well as short-term history as well as current execution performance. Service evaluators provide ranks and also determine performance trends that may be used to select services.

We have integrated a partner relationship evaluation methodology, a Log-based performance evaluation component and a service-price component in our prototype system. The Partner relationship evaluation methodology (refered to as PRM module) assigns *rankings* and *ratings* to invidivual business partners or CPs based on long-term conformances to contractual agreements and some other performance factors (explained below). The Log-based evaluation component analyzes transaction logs and provides classified lists of CPs based on classification requirements in metarules. The service-price component provides service prices per invocation.

B.1 Partner Relationship Evaluation

The PRM module implementation periodically evaluates 3rd party CP web services with respect to contractual agreements. The evaluation engine of the PRM system makes use of several factors in the evaluation process of the CPs. The list of factors include real-time (online) factors such as *service pause* and *response time* and various offline factors such as *complain processing time* and *user satisfaction*. The input data comes from various sources and formats and has to be preprocessed before storing it into a knowledge base. The evaluation framework uses this knowledge base.

The PRM module allows the SP to evaluate the CP on an as-needed basis. Thus evaluation can be periodic as well as manual. For example, if the evaluation was being done on a monthly basis and on receiving large number of complaints from the subscriber about service quality deterioration, the SP can undertake to evaluate the CPs immediately rather than wait for the monthly evaluation.

The frequency of partner's evaluation update can be either in hours, days or months. Thus the offline evaluation of business process must be synchronized with this frequency. Let the offline evaluation frequency be fo and the PRM partner evaluation frequency be f_p , then
$f_p = N f_o$, where N is a positive integer and $N > 1$
The above equation signifies that the offline evaluation can be as frequent as the PRM evaluation but not less frequent than that of PRM. If the offline evaluation is less frequent than PRM it means that the system is not using the current partner ratings and rankings but the obsolete one. Currently, the offline evaluation frequency is equal to that of the PRM evaluation. Whenever PRM completes the evaluation of any of the business process the offline evaluation is triggered manually.

B.2 Log-based Performance Evaluation of CPs

The Log Miner uses Generalized Disjunctive Association Rules (GDAR) [10] and stores the perfor-

mance (instance level performance) results of the various content providers' actual transactions. It is also referred to as *GDAR module* from the technique used in it. Unlike PRM, GDAR module only records technical performance factors like response time, availability, latency etc. Given a context (e.g. find services that were used by *premium* class users and had a *fast* response), GDAR produces a set of disjunctive association rules that satisfy the context and has a certain degree of *confidence. Confidence* is a measure of how frequently the rule has been observed in the logs. For example, a certain output from GDAR might look like:

premium and fast=> PVR (Gurgaon) & PVR (Saket) & Citibank (New Delhi). Conf=0.9

This implies that PVR (Gurgaon) and PVR (Saket) and Citibank (New Delhi) have individually been invoked by the service provider for *premium* class users and have yielded *fast* response, with a confidence of 0.9. Offline module uses such rules to filter out services as prescribed in the metarules. Currently, the measurement factors are based on categorical variables (e.g. userType is a categorical variable having 3 values: premium, mid-segment and regular). However, non-categorical variables can be transformed to categorical variables with a defined mapping function.

Because of PRM and GDAR's characteristic to remember the business partner's performance history, they act as a feedback control loop in the ODSS-in-ENDs. Each time the offline algorithm is executed it gets feedback from PRM and GDAR about a business partner's performance and this is reflected in the new rankings of the alternative workflows in the *context-dependent alternative workflow* store.

C. WorFare Engine (WORkflow Filtering And Ranking Engine)

The WorFare engine is analogous to a CPU (Central Processing Unit) and fetches data from external modules like Log Miner, Partner Relationship Module and SP Business Policies to process. It has access to all the required business policies (metarules) as well as the workflow templates and instances for a service. There is an *accessor* and a *transformer* for each service evaluation module the WorFare engine is connected to. For each metarule in the system and for each service supported by the service provider, the WorFare engine performs the task of evaluating each workflow instance of a service and selecting a subset of workflows that pass the evaluation criteria provided in the metarules. The engine also contains a metarule parser that parses the rule instances and determines the appropriate evaluation criteria to be applied on service evaluation data from the various modules. For example, a rule might formulate that

```
OFFLINE:
For each meta-rule of a task T in the WorFare_Engine
    GDAR_context = Get_GDAR_context (T_Meta-rule)
    GDAR_Output[] = GDAR (GDAR_context)           /* A number of WS */
    PRM_type = Get_PRM_Info (T_Meta-rule)         /* Rating or Ranking */
    PRM = Get_PRM (T_Meta-rule)
    Price = Get_Price (T_Meta-rule)

FILTER_WF:  For each WorkFlow[i] of a task T in the WorkFlow_Base   /* WorkFlow is an array of WS */
            /* SubSet interface determines if the component web services in the i^th workflow are a subset of
               those generated by the GDAR */
            If SubSet ( WorkFlow[i], GDAR_Output[] ) is FALSE
                /* Discard this WorkFlow */
                WorkFlow[i] = NULL
                GOTO  FILTER_WF
            For each j^th WS_Component in WorkFlow[i]
                If PRM_type == "Rating"
                    If PRM_RelOp == "<" or PRM_RelOp == "≤"
                        If PRM < PRM_WebService_Rating (WS_Component[j])
                        /* WS does not meet the PRM Rating requirements */
                        /* Discard this WorkFlow */
                        WorkFlow[i] = NULL
                        GOTO  FILTER_WF
                    If PRM_RelOp == ">" or PRM_RelOp == "≥"
                        If PRM > PRM_WebService_Rating (WS_Component[j])
                        /* WS does not meet the PRM Rating requirements */
                        /* Discard this WorkFlow */
                        WorkFlow[i] = NULL
                        GOTO  FILTER_WF
                If PRM_type == "Ranking"
                    If PRM_RelOp == "<" or PRM_RelOp == "≤"
                        If PRM < PRM_WebService_Ranking ( WS_Component[j] )
                        /* WS does not meet the PRM Ranking requirements */
                        /* Discard this WorkFlow */
                        WorkFlow[i] = NULL
                        GOTO  FILTER_WF
                    If PRM_RelOp == ">" or PRM_RelOp == "≥"
                        If PRM > PRM_WebService_Ranking ( WS_Component[j] )
                        /* WS does not meet the PRM Ranking requirements */
                        /* Discard this WorkFlow */
                        WorkFlow[i] = NULL
                        GOTO  FILTER_WF
/* Now we have a set of workflows which have passed all requirements viz. GDARs and PRM */
/* We create an ordered list of these workflows using some ranking criteria */
    Rank_WF ( WorkFlow[] )   /* Orders the surviving workflows in WorkFlow[] */

/* Construct the rule and add to the Rulebase */
    Construct_RuleBase_LHS ( T_Meta-rule )
    Construct_RuleBase_RHS ( WorkFlow[], T_Meta-rule )
```

Fig. 4. Pseudo Code of Offline Module

all *premium* users will use services that have a PRM rank less than 2. These filtering criteria are used consecutively on the set of available workflows to select a set of *alternative workflows*.

Alternative workflows are then ranked by the engine and these ranked workflows are stored in the *context-dependent alternative workflow* store. Workflows may be ranked based on several criteria (e.g. least costly, fastest response etc). However, the service provider may also provide its own ranking criteria by supplying its own ranking algorithm. The WorFare engine offers easy pluggability of such ranking algorithms by exporting an API to service providers to implement such algorithms. Figure 4 shows the pseudo code of the offline module.

The *accessors and transformer* of each independent evaluation component (viz. PRM, Log Miner, SP policy module and the databases storing business processes) retrieves and transcodes the information provided by the components in a format understandable by the engine. This makes the system implementation flexible and extensible to be able to add new evaluation modules in future. When a new evaluation module needs to be plugged to the engine, one can write an accessor to transcode the information provided by the module and the engine implementation need not completely change; though it needs to incorporate this new evaluation information in ranking of the workflows.

D. Pluggable Evaluation Modules

The use of accessor and transformer modules in ODSS-in-ENDS to connect the WorFare engine to any service evaluation module enables a pluggable architecture. Some evaluation modules (PRM, GDAR) are intrinsic to the system and are tightly integrated. Other evaluation modules might be provided by ser-
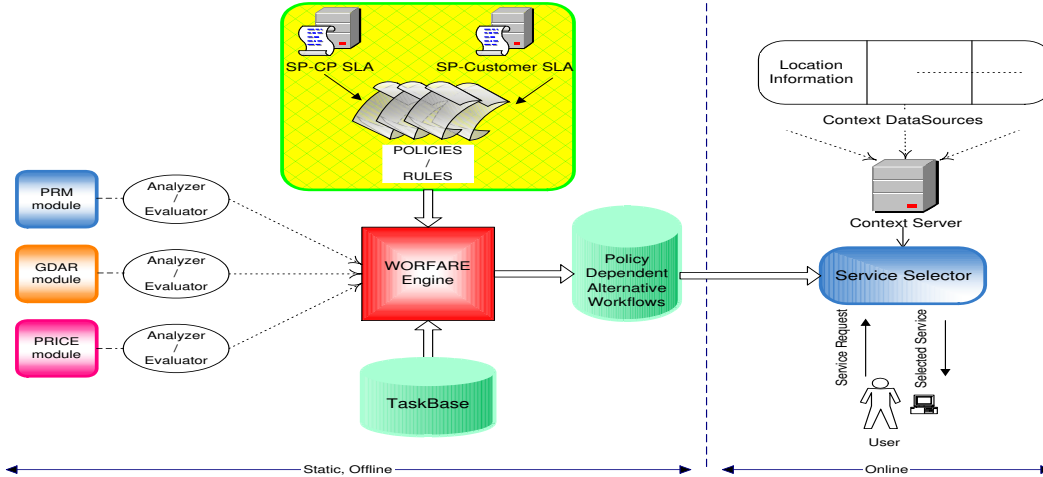
Fig. 3. ODSS-in-ENDs Service Evaluation and Selection Platform

vice providers who want their own in-house evaluation algorithm to be plugged in to the system. Plugging new evaluation module into the ODSS engine requires writing of an additional accessor and transformer. One can write few lines of code extending the engine to incorporate the new evaluation module in the ranking of the business processes. For example, we have plugged in a Price module to filter workflows and services based on pricing preferences.

E. Online Service Selection Module

Online module selects one of the pre-ranked workflows to deliver a particular service (or task) upon user request. A user query usually specifies the task requested, followed by some required input parameters. Context information (user location, network status etc) is captured by the Online module through user registration information available to it. The context information is used along with the service ID to retrieve a set of alternative workflows from the *context-dependent alternate workflow* store.

We have integrated a context aggregator [8] that provides us with run-time performance indices of several content provider services in the system. Metarules often have run-time context check requirements (e.g. the current response time of the service should be *fast*). These run-time checks are executed by the Online module for each of the services belonging to the ranked workflows. The first workflow that passes through the test is executed to deliver the particular service. Online contains a Java/SOAP client to invoke these processes using the standard SOAP/HTTP APIs. When the Online module selects a workflow to be executed, the corresponding process signature (parameters to be passed to this process so as to invoke it) is retrieved from the request sent by the user and the process is invoked by the SOAP client with

ONLINE:

/* Form the current context using the fundamental variables */

Current_Context = Get_Current_Context ()

SELECT_WF: For each Rule in the Rulebase
 /* Check if the _c fundamental variab les of the LHS of the Rules in the Rulebase are
    triggered by the current context */
              If Match_LHS_Rules (Rule, Current_Context) is TRUE
               For each i [th] WorkFlow associated with the Rule
                For each j [th] WS_Component in the WorkFlow[i]
                 /* Query Context S erver to confirm that each WS
                    component meets the desired constraints or
                    provides the requisite Quality of Service   */
                  If ContextServer_Verify (WS_Component[j], Rule) is FALSE
                   GOTO SELECT_WF
                 WBISF_Execute (WorkFlow[i])   /* End of Online Algorithm */

Fig. 5. Pseudo code of the Online service selection module

the requisite arguments.

Faults in the system could happen due to failure of any service in the workflow. This is detected by the Online module and an alternative workflow is selected. This increases the reliability of our system. Pre-ranking of the workflows ensures that such reliability of service delivery is provided with the least possible digression from the desired optimal service execution performance bounds. Figure 5 shows the pseudo code of the Online module.

IV. PROTOTYPE DEMONSTRATION

We have developed a prototype demonstration of ODSS-in-ENDs to illustrate the capabilities of our system. Our prototype demonstration implements a *movie ticket reservation* service for cellular users. Screendumps shown in this paper are for the web-based interface to our system. Please note that our

entire system is also accessible as a web service and also through voice (through a telephone number). We consider several workflow templates and workflow instances to execute the task. In the template used in this paper, we assume that in order to execute the service, a movie ticket provider site has to be integrated with an online bank account. We consider 4 different movie ticket providers and 2 online banking services. As such, there are 8 instances of workflow instances (consider all combinations) that ODSS-in-ENDs could select from.

To demonstrate the enablement of efficient service selection, we assume that the user is agnostic towards any particular theatre location (within a particular geographic domain). We have 3 users registered to our system. We associate *user category (premium, midsegment, regular)*, *location (remote, local)* and *networkstatus (peak time, offpeak time* as the context of each user. The metarules that specify the context-based selection criteria in our demonstration are shown in Figure 6.

1. *premium & peak & remote => History(peak & fast) & PRM (Rank < 4) & Price (<,0.75$) & run-time-context (responsive==fast)*

2. *premium & offpeak => History (offpeak & fast) & PRM (Rank < 3) & run-time-context (responsiveness=medium,fast)*

Fig. 6. Metarules used in the demonstration of *Movie Ticket Reservation* value-added service

The workflow instances as well as service charge related information (charge per invocation) of the component services are shown in table I.

The WorFare engine initially selects and ranks the alternative workflows with respect to total service charge per invocation. The selection process considers PRM as well as evaluation from the GDAR module. From the first metarule in Figure 2, we use the GDAR module to determine services that are *fast* and have been used at *peak* hours. This yields the following result:
Waves (Noida);PVR Movies(Gurgaon);PVR Movies (Saket);HDFC (SDA branch); ICICI (CP branch); Yahoo Map Service; DHL Courier Service)
WorFare engine produces a set of 5 workflows that satisfy all the evaluation criteria. Figure 7 shows our web-based front-end to ODSS-in-ENDs where a *premium* user requests for a movie ticket reservation service at *peak* hours. The output produced by ODSS-in-ENDs along with the execution logs of the Online module are shown in Figure 8. The Online module initially retrieves the set of executable workflows from the *context-dependent alternative workflow* store. Thereafter, it performs run-time checks on each component service (as prescribed by the metarule
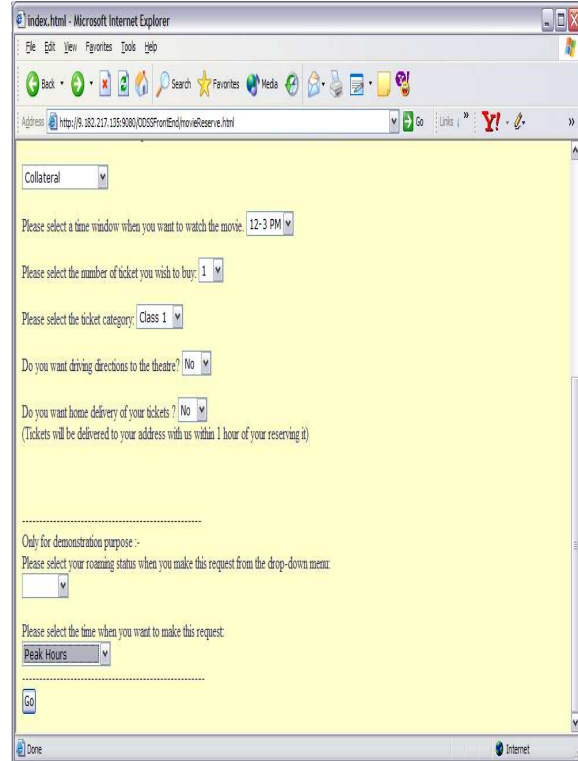

Fig. 7. Web-based Front-end to accept user requests for movie ticket reservation service

in Figure 2) using the Context Aggregator. Logs of the workflow execution show how the Online module manages the selection. Figure 9 shows the execution history when the same user requests the service at *offpeak* hours. We observe that a different workflow instance was selected for execution this time as the request was made during offpeak hours. Our demonstration also includes other workflow templates such as "Reserve a movie ticket and obtain driving directions to the theatre".

ODSS-in-ENDs runs on the WebSphere Business Integration Server Foundation (WBISF) application server platform. IBM WBISF provides support for deploying BPEL processes and web service-based component applications. The BPEL workflows, deployed on the Business Process Execution (BPE) container in the WBISF, are generated by the WSADIE with SOAP/HTTP bindings. The BPE container is a specialized J2EE application that executes business processes and flows.

V. RELATED WORK

Service delivery platforms utilize various standards/APIs [12], [2], [9] to develop in-house services that utilize 3rd party content provider services/platforms. The core functionality of such platforms/APIs is to enhance the creation and deployment of services in enterprise networks. In [6], the authors define an

Movie ticket providers	PVR (Saket), PVR (Gurgaon), Chanakya theatres, Waves (Noida)
Online banking services	HDFC (SDA branch), ICICI (CP branch)
PVR Movies(Saket)	0.55$
PVR Movies(Gurgaon)	0.60$
Chanakya Theaters	0.25$
Waves (Noida)	0.50$
HDFC (SDA branch)	0.40$
ICICI (CP branch)	0.55$

TABLE I
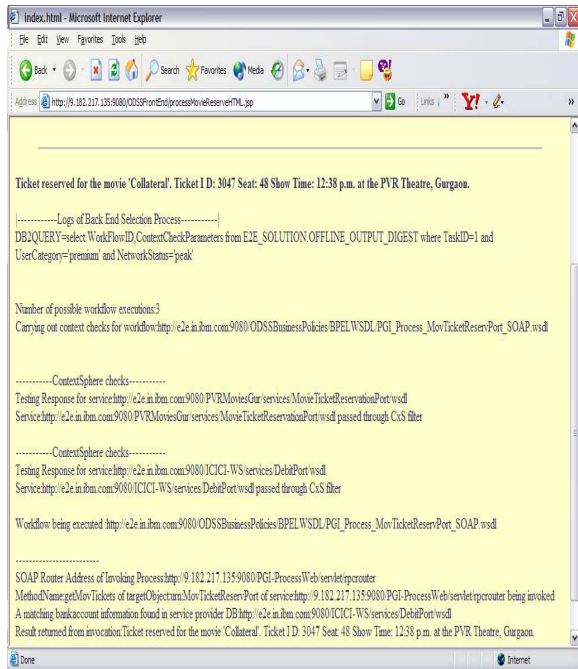
DEMONSTRATION DATA USED BY OUR PROTOTYPE IMPLEMENTATION

Fig. 8. Result returned from ODSS-in-ENDs along with execution logs

Fig. 9. Execution logs when request is made at *offpeak* hours

intelligent-agent based Service Peering and Aggregation Server (SPAS) architecture. The main purpose of SPAS is to allow easy aggregation of content in the Service Provider's domain as a peer. SPAS also uses a repository of user profiles (services ranked by the *subcribers*) for runtime conflict resolution for particular subscriber. SPAS thus relies on static rankings of services by the subscriber. However, to the best of our knowledge, none of these platforms are adaptive to changing business strategies (business context) while supporting dynamic evaluation and selection of network content providers (and their services) based on context.

Various standards for inspection and easy deployment of web services [1], [3], [4] have been proposed and are being adopted by service and content providers or business services in general.
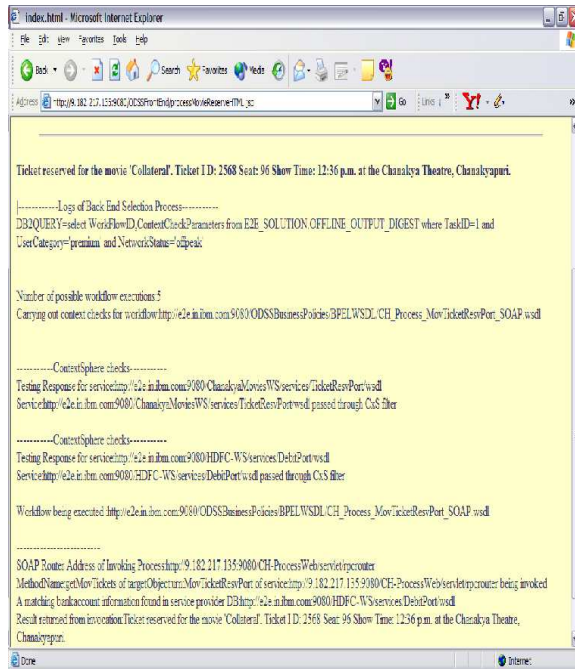
Our architecture addresses evaluation of web services representing content providers. Evaluation is based on performance and changing business strategies. There is a body of work on dynamic composition [11] of web services at runtime. However, they do not concern themselves with dynamic selection amongst competing web services, or on the use of business rules to arbitrate selection among the competitor services.

VI. CONCLUSIONS

We present the design and implementation of an integrated architecture for efficient evaluation and selection of 3rd party web services for service providers. Our system integrates several evaluation methodologies and considers changing business contracts, performance trends of the services and user profiles while selecting a service. We are working towards integrat-

ing user preferences as well as privacy considerations and personalization in our system.

REFERENCES

[1] Web Services Description Language 1.1. World Wide Web, `http://www.w3.org/TR/wsdl12`.

[2] Application Programming Interface 3GPP, Open Services Architecture. 3G TR 29.998, `http://www.3gpp.org`.

[3] Keith Ballinger, Peter Brittenham, Ashok Malhotra, William A. Nagy, and Stefan Pharies. Web Services Inspection Language 1.0, `http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.h%tml`.

[4] Universal Description Discovery and Integration Specification. World Wide Web, `http://www.uddi.org`.

[5] K. Erol, J. Hendler, and D. Nau. HTN planning: Complexity and expressivity. In *Proc. AAAI.*, 1994.

[6] B. Falchuk, K. E. Cheng, F. J. Lin, B. Pinheiro, , and V. Jokubaitis. An agile server for cross-provider service peering and aggregation. In *IEEE Communications*, pages 126–136, March 2003.

[7] BPEL4WS. Business Process Execution Language for Web Services. World Wide Web. `http://www-106.ibm.com/developerworks/library/ws-bpel/`, 2003.

[8] Hui Lei, Daby M. Sow, John S. Davis II, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *Mobile Computing and Communications Review (MC2R)*, 6(4):45–55, October 2002.

[9] A.-J. Moerdijk and L. Klostermann. Opening the networks with parlay/osa: Standards and aspects behind the apis. In *IEEE Network no. 3,*, pages 58–64, May 2003.

[10] Amit Nanavati, Krishna P. Chitrapura, Sachindra Joshi, and Raghu Krishnapuram. Mining generalised disjunctive association rules. In *10th International Conference on Information and Knowledge Management (CIKM), Atlanta, Georgia*, pages 482–489, 2001.

[11] Mangala Gowri Nanda and Neeran Karnik. Synchronization analysis for decentralizing composite web services. In *ACM Symposium on Applied Computing, Melbourne, Florida*, pages 407–414, 2003.

[12] Parlay API Specification Parlay Group. `http://www.parlay.org`.

[13] X. Xiao and L. M. Ni. Internet qos: A big picture. *IEEE Network magazine*, 13:8–18, March 1999.