Web Service Competition:
A New Approach to Service Selection

Mehran Najafi* , Kamran Sartipi**, and Norman Archer***
* Global Business Services, IBM Canada
** Engineering and Applied Science, University of Ontario
*** DeGroote School of Business, McMaster University

As the number of web services that offer similar functionality increases, more sophisticated techniques for service discovery and selection will be needed. Traditional approaches compare web services based on their description published in service registries which include QoS and price/performance ratios, as well as adaptability. This information is generated by the service developer and may not be fully trustable by the client. Moreover, alternative services perform differently in different client contexts that cannot be determined accurately by service descriptors. In this paper, we propose a novel service selection approach that compares alternative services based on their performance in a specific client context. For this purpose, we extend the SOA infrastructure model through a component named the competition desk, that holds a competition among alternative services taken from the service client. As a result, clients can choose the service that works best for their needs.

1 Introduction

Service-Oriented Architecture (SOA) [9] is a high-level and technology-independent concept that provides architectural blueprints for enterprise systems. SOA based architectures focus on dividing the enterprise application layer, where its components (as services) have a direct relationship with the business functionality of the enterprise.

Emerging technologies such as cloud computing [16] propose to provide applications as services which can then be distributed across the network and reused in other applications. Consequently, the number of web services that offer similar functionality increases, and discovering relevant services (service discovery) and choosing the best one to meet the client's needs (service selection) will be more challenging.

On the other hand, the Smart Internet [14] was recently introduced as one of the candidates for the next Internet generation, where the Internet is moving toward a more user-centric network. The Smart Internet will allow clients to select services that work best, based on client contexts and needs. Toward these goals, the underlying conceptual model and infrastructure of SOA must be extended and modified to meet the new requirements.

Based on the traditional SOA model, service providers publish descriptions of their services in the service registry that is used by service clients to discover and select services. This model suffers from the following limitations.

- Service descriptions that are provided by service providers may not be trustable or accurate enough.
- Service descriptions are usually expressed globally while service features such as performance and accuracy are different for different clients, depending on their needs and contexts.
- Less well-known services are not given an opportunity to show their features.

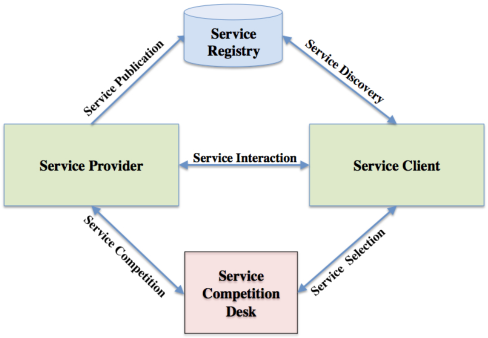


Figure 1: Proposed extended SOA infrastructure model.

- Service features vary with different measures and are obtained under different situations. Therefore they cannot be simply and fairly compared, based only on their descriptions.

Inspired by the business domain, where in many cases service providers compete to win clients, we propose a service selection approach based on service competition (*WS-Competition*). In other words, while the traditional passive service selection relies on service description comparisons, the proposed active service competition compares services based on their performance for a specific client application.

We propose to extend the traditional SOA infrastructure model by adding a new component that we call the *Competition Desk* (Figure 1). It enables a service client to submit a list of candidate services (competitors) as well as a set of test cases and competition policies. The competition desk asks each of the candidate services to employ a representative and then it holds a competition among the service representatives and returns the competition results to the client (Figure 2). As a result, the client can choose the service that works best for its needs. The competition desk supports both data and task services. While a data service processes the client's data at the server side, a generic and client-side software agent (the service representative), is employed by a task service to process client data at the client side and deliver the service response locally to the client.

The organization of this paper is as follows. Data and task services are introduced in Section 2. Service selection based on the proposed competition model is described in Section 3. Two case studies using a developed prototype system are presented in Section 4. The discussion (Section 5) explores the applications and challenges posed by the web service competition approach. Section 6 discusses work related to our approach. Finally, conclusions and future work are provided in Section 7.

2 Data and Task Services

An enterprise system needs to provide different types of web services to model actual services in the business domain. Each web service executes one or more enterprise business processes where a business process applies business rules and performs business actions on internal (server-side) and external (client-side) business data in a defined order. Therefore, a service can be modeled by a collection of business components which includes business processes, rules, actions, and data. Moreover, a business process can have server-side and/or client-side processing. As opposed to the traditional SOA that considers business services to be identical, in [12] and [13] we proposed to categorize business services into either data or task services.

2.1 Data Service

This represents a typical web service in the current SOA model where the service processes the client's data and resources completely at the server site. Consequently, a data service includes only server-side processing and returns a service response in the form of data that will be consumed directly by the client. In other words, a data web service (DWS) is modeled by a function that receives the required remote service parameters (RSP_{DWS}) from the service client and returns the resulting remote service responses (RSR_{DWS}) to the service client, as:

$$DWS : RSP_{DWS} \longrightarrow RSR_{DWS}$$ [1]

[1] In defining a function (e.g., DWS), we use the sets of "parameters" (e.g., RSP_{DWS}) and "responses" (e.g., RSR_{DWS}) to represent the types of the function's input and output.

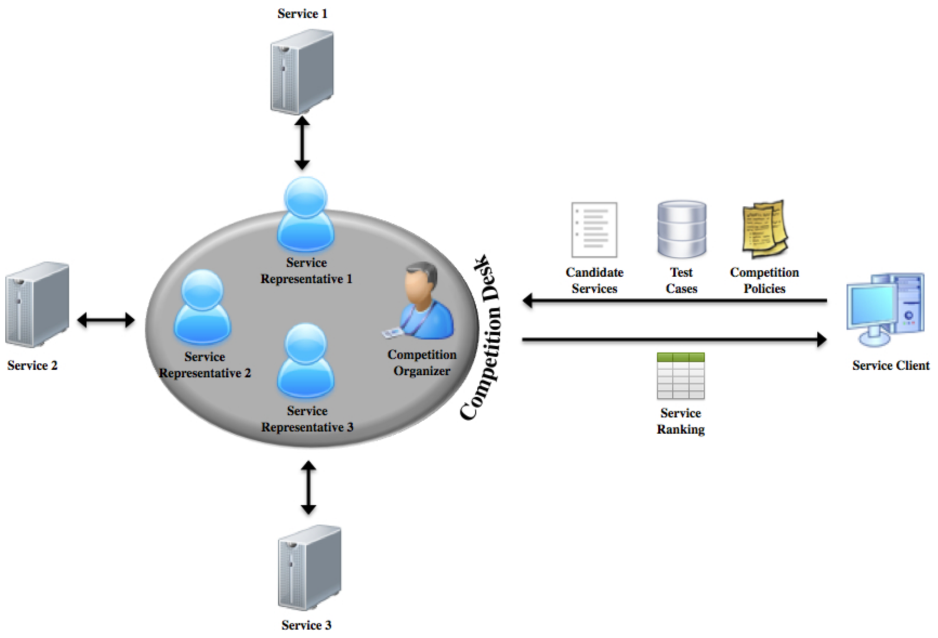


Figure 2: High level view of the proposed web service competition.

2.2 Task Service

The concept of task services is introduced in [13] as web service with the capability of processing the client's data and resources partially or completely at the client platform. A task service performs the required server-side processing and then it defines a *task* including the client-side processing to be performed by the *Service Representative (SR)* at the client platform. A service representative is a generic client-side software agent with a built-in process engine that can be employed by different service providers to perform different task services. The service representative requires both task logic and task data to perform client-side processing that can be provided from different sources. Based on the proposed service model, while a data service returns business data as its service response, a task service returns a task message with the following components:

$$Task =< Model, Knowledge, Data >$$

- *Task Model* specifies an abstract Business Process Model (BPM) representing the required client-side processing.
- *Task Knowledge* provides the required Business Rules (BR), i.e., descriptive knowledge, and Business Actions (BA), i.e., procedural knowledge, to realize the specified abstract BPM.
- *Task Data* represents the server-side Business Data (BD) that are consumed by the business rules and actions during the business process. However, the required client-side business data are provided locally by the service client.

Task components are messages that can be transmitted efficiently over the network. The service representative uses the received task components to perform client-side processing of the task service on the *local service parameters* to provide *local service responses*. Consequently, a task web service (TWS) is modeled by a function where it receives the required remote service parameters (RSP_{TWS}) from the service client and then it returns the resulting task message ($Task_{TWS}$) to the service representative (SR_{TWS}) to process the local service parameters (LSP_{TWS}) and generate the local service responses (LSR_{TWS}).

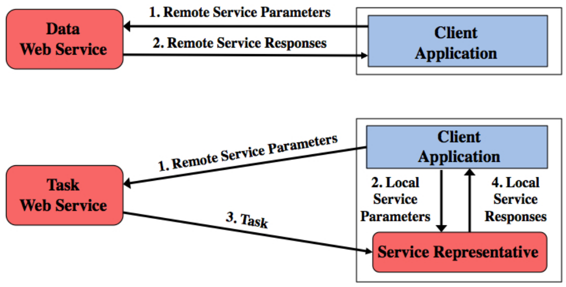

Data Web Service
1. Remote Service Parameters
2. Remote Service Responses
Client Application
Task Web Service
1. Remote Service Parameters
3. Task
Client Application
2. Local Service Parameters
4. Local Service Responses
Service Representative

Figure 3: Comparison of Data and Task Services.

$$
\begin{aligned}
&TWS : RSP_{TWS} \longrightarrow Task_{TWS}\\
&SR_{TWS} : Task_{TWS} \times LSP_{TWS} \longrightarrow LSR_{TWS}
\end{aligned}
$$

Task service applications include client data that should remain at the client platform since they cannot be transmitted to the server for reasons of confidentiality, real time response requirements, or being too large to transmit efficiently. Figure 3 compares these two types of services and the following example illustrates the different applications of data and task services.

- A typical financial adviser data service asks for the client's financial information to provide personalized advice.
- A financial adviser task service generates a set of general financial advice (e.g., stock buy and sell advice) according to the client's preferences (server-side processing). Moreover, it generates a task message for the service representative including the required procedure (task model) and guidelines (task knowledge) to customize the general financial advice (task data) based on the client's personal information (client-side processing).

Finally, the following issues should be mentioned about task services.

1. In order to invoke a task service, the service client is required to install and use the generic service representative which may increase client-side complexity. However, the developed service representative prototype, which is introduced in [12], requires small memory (a few MBytes) and offers reasonable computational speed. Moreover, the two types of services offer a tradeoff between client-side complexity and the importance of local processing of the client's data.

2. Instead of customizing the local and generic service representative, a service provider can send a customized mobile agent to the client to perform the client-side processing of its web service. However, mobile agents may introduce security and privacy challenges and they may increase network traffic.

3. Task services have advantages over other client-side processing techniques (such as scripting or Rich Internet Applications) due to their composability and scalability.

4. In addition to preserving the client privacy, task services increase client security since the client has control of the required computer resources (e.g., CPU time, storage, and memory) for the service representative.

5. Required knowledge for the service representative can be enterprise assets and resources, and revealing them may violate the enterprise's privacy. To prevent this security vulnerability, a service provider can use one of the following techniques. (I) Enterprise knowledge can be divided to be applied locally (at the provider side) by the service or externally (at the client side) by the service representatives. Therefore, the critical knowledge (e.g, market analysis) remains at the service provider, while the non-critical knowledge (e.g., advice customization guidelines) is sent to the service representative. (II) The service client only receives the service response from the service representative. Consequently, the service client does not have access to the knowledge transferred between the service provider and its representative. Moreover, encryption techniques can be used for data transmissions between a service provider and representative to improve enterprise security.

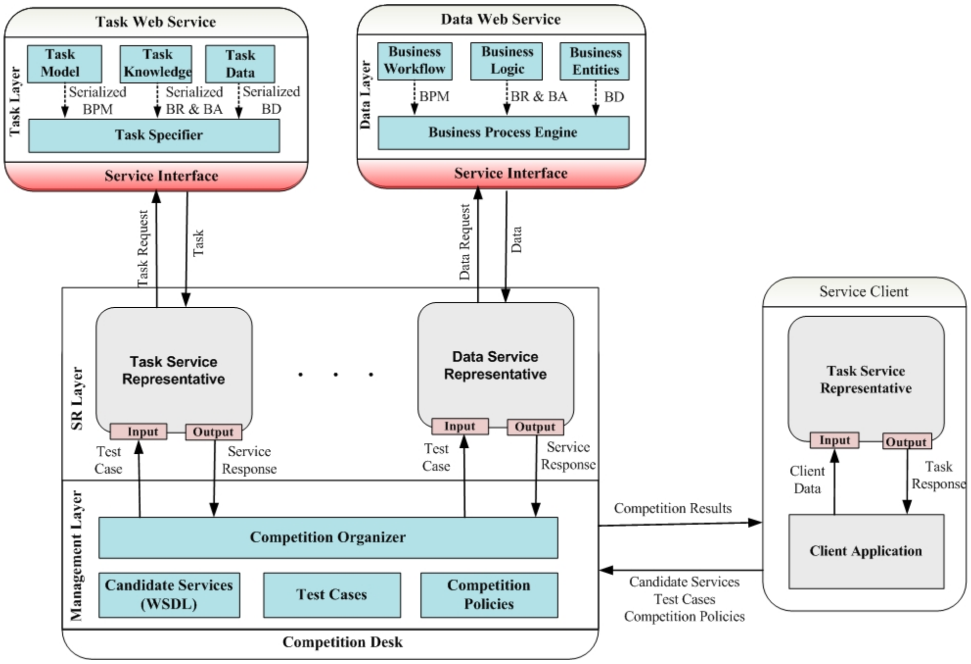


Figure 4: Architecture for the proposed web service competition. The competition desk holds a competition among the service representatives of the candidate services using the test cases and policies submitted by the client.

The service representative structure is proposed in the next Section, and task service examples are presented in Section IV.

3 Web Service Competition

In order to perform a service selection based on the proposed service competition model, an architecture is required. The proposed architecture (Figure 4) includes three main components as follows.

3.1 Service Client

A service client consists of a traditional client application that sends a data or task service request to a service provider. In order to receive task services, a service client is equipped with a generic service representative. Consequently the client application receives the service response directly from the service provider (data services) or indirectly from the service representative (task services).

To select a service that best matches with the client's need, a service client first searches for candidate services from the service registry. When these have been found, the service client submits the following information to the competition desk:

1. **Candidate Services** which are described by WSDL (Web Service Description Language) documents obtained from the service registry.

2. **Test Cases** where each test case is represented by a couple (*Service Parameters*, *Expected Results*). For a data service, *Service Parameters* include the remote service parameters and *Expected Response* is the expected remote service response. However, a task service requires both remote and local service parameters as the *Service Parameters* as well as the corresponding local service response as the *Expected Response*.

3. **Competition Policies** which define the relevant competition factors such as accu-

racy, cost, response time and other performance metrics. Each of these factors is associated with an evaluation function to guide the competition organizer to obtain their values. Finally, it includes a ranking policy to rank services based on their results from the competition. The ranking policy determines the weight of each factor in the service ranking.

3.2 Services Provider

Corresponding to the type of service, each service provider has either a data or task layer where the former performs the server-side processing of a web service and the latter defines a task for the service representative to perform client-side processing at the client side. A hybrid service that includes both client-side and server-side processing will require both data and task layers. Therefore, the enterprise business components are divided between these two layers. While the data layer applies the business components, the task layer sends the business components to the client-side to be applied by the service representative.

Service Interface. This component supports the communication contracts (message-based communication, formats, protocols, security, exceptions, and etc) for the services.

Data Layer. Server-side business processes, rules and actions, and data are stored in the business workflow, logic, and entity components, respectively. The business process engine executes the corresponding business process with each service. This layer responds to the client with a single-segment response message (a data message).

Task Layer. Client-side business processes, rules and actions, and data are stored in the task model, knowledge, and data components, respectively. Since business components in this layer are sent to the client side, they must be serializable. The task specifier provides the required model, knowledge, and data for each task request to be sent by a three-segment response message (the task message) to the client.

3.3 Competition Desk

The competition desk manages the web service competition and has two layers: a service representative layer and a management layer, as follows.

Service Representative (SR) Layer. This layer includes the service representatives of the candidate services. A service representative is a generic software agent that represents a service at the competition desk (or client side) where it has different components for task and data services, as follows.

- *Task Service Representative*: This agent, shown in Figure 5, performs a task with the following components.

  - *Service Stub* : invokes a task service by sending its remote service parameters to receive a task message.

  - *Input*: inputs local service parameters of each test case from the competition organizer.

  - *Knowledge Base*: stores basic and internal business rules and actions to relieve the service provider from sending them each time. Using this internal knowledge base, a service representative can be customized for each domain.

  - *Business Process Engine*: executes a task instance by applying business rules and performing business actions on business data.

  - *Task Manager*: provides the following functionalities to support the entire life cycle of a task instance (i.e., from creation to termination).

  1. *Task Invocation*: calls a task service through the service stub to receive a task message.
  2. *Task Instantiation*: creates an abstract business process based on the task model and then realizes the abstract process using internal and external task knowledge to generate a task instance.
  3. *Task Execution*: passes the task instance with the relevant business

data (i.e., received from the input component and task data segment) to the business process engine to be executed. Finally, the task manager sends the task results to the output.

- *Local Memory*: stores the process variables of the task process during the task execution.

- *Output*: outputs task service response for each test case to the competition organizer.

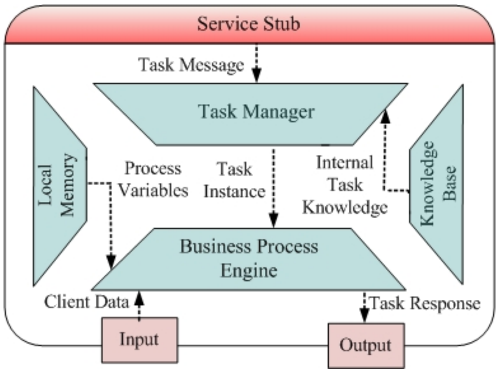


Figure 5: Task service representative components.

- *Data Service Representative*. This agent, shown in Figure 6, is assigned to a data service and invokes it for each test case using the following components.

  - *Service Stub* : invokes the data service for each test case to receive the corresponding service response.

  - *Input*: inputs service parameters of each test case from the competition organizer.

  - *Output*: outputs data service response to the competition organizer.

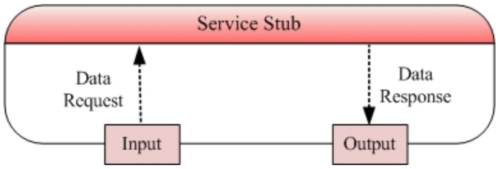


Figure 6: Data service representative components.

Management Layer. The competition organizer sets up a service competition based on the information received from the client, executed in the following four phases.

1. *SR Instantiation Phase*: creates one generic Service Representative (SR) for each candidate service. Then, it configures each SR's service stub using the WSDL description (which specifies a service address and port) of the corresponding service. When this phase is completed, there is an assigned SR for each candidate service that can communicate with the service.

2. *SR Initialization Phase*: calls the task manager of each task service representative to invoke its task from the assigned service and then to instantiate the task. There is no initialization phase for data service representatives.

3. *Competition Phase*: passes the test cases one by one to each SR and waits until they respond. During the competition, it collects information regarding the specified competition factors (such as response times). When the competition organizer receives all the service responses, it computes the value of each competition factor for each candidate service, based on the evaluation functions specified by the client.

4. *Ranking Phase*: applies the ranking policy to the obtained results to rank the candidate services.

This layer also contains internal components storing competition policies, test cases, and candidate services in different categories that can be used by clients when they cannot or do not intend to provide this information.

4 Case Studies

To evaluate the effectiveness and feasibility of service selection based on the proposed web service competition approach, we developed a prototype system of the proposed architecture including the competition desk, data and task

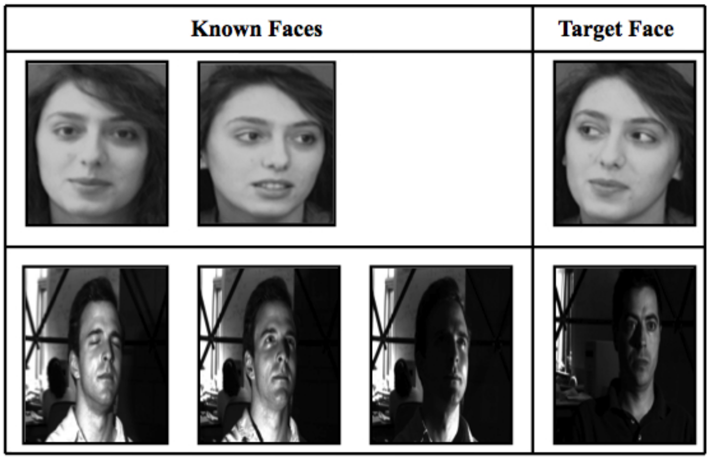


Figure 7: Two test cases for the data service competition: (top) a test case in experiment one with the expected result = "Yes" ; (bottom) a test case in experiment three with the expected result = "No". While the former represents the client's interest in face recognition services which recognize facial orientation, in the latter, the client searches for a face recognition service which works under variable illuminations.

service providers, data and task service representatives, and service client. This prototype, *WS-Competition version 1.0*, is based on *J2EE* 1.5 technologies and the *Apache Tomcat* 6.0 application server. Task service representatives use *Drool* version 5.0 as their business process engine to execute task instances. Moreover, the competition desk has an internal *MySQL* database to store test cases. Business process models are converted to XML format; business rules are expressed by production rules and encoded by *PMML* standard version 3 [15]; business actions are defined by Java statements or functions; and business data are defined by Java beans and serialized to form task messages. We ran two web service competition scenarios among data and task services as follows.

4.1 Data Service Competition

A client searches for a face recognition web service that verifies whether facial images belong to the same person. Although there are several algorithms for face recognition, there is no evidence to show that any one of them is consistently the best one under all circumstances (e.g., different lighting and orientation conditions) [6].

Candidate Services. Each service takes a number of facial images belonging to the same person as well as an unknown facial image (target face). Then it verifies whether the target face represents that person. This result is associated with a confidence level that ranges between 0 and 1, where 0 represents zero confidence that a match has been made, and 1 represents full confidence in the match.

WS (face 1, $\cdots$, target face) = (Yes / No, confidence)

In this case study, the candidate services include five face recognition services respectively based on *Principal Component Analysis* (PCA), *Independent Component Analysis* (ICA), *Linear Discriminant Analysis* (LDA), *Support Vector Machine* (SVM), and *Incremental Neural Network* (NN) techniques (these techniques are introduced in [6]). In addition to a recognition technique, a face recognition service requires a data set including training facial images. To verify the target face, each service adds the known faces to its training set; then it builds or completes its recognition model; and finally the model is applied to the target face.

Test Cases. Each test case is represented in the form of *(face 1, . . . , target face, expected response)* where the expected response is either "Yes" or "No". The service client chooses test cases that have similar conditions to the actual cases that he/she wants to have recognized.

Competition Policies. The client specifies the competition factors as *Accuracy*, *Confidence*, and response *Time* with the following evaluation functions where S and F represent the set of successful and failure test cases, respectively, for a service and $\mid A \mid$ represents the cardinality of set A.

$$Accuracy = \frac{\mid S \mid}{\mid Test\ Cases \mid}$$
$$Confidence = \frac{\sum_{i\epsilon S} confidence_i - \sum_{i\epsilon F} confidence_i}{\mid Test\ Cases \mid}$$
$$Time = Average(Response\ Time)$$

Finally, the client assigns 0.5, 0.3, and 0.2 for the weights of the *Accuracy*, *Confidence*, and *Time* in the ranking policy.

Experimental Results. We ran three experiments with the prototype system to discover the best candidate services in each specific condition. *ORL* and *Yale* face data sets were used in these experiments. *ORL* includes 400 facial images of 10 persons while *Yale* has 165 facial images of 15 persons. Facial images were divided into three sets: (1) training data, (2) competition data, and (3) evaluation data, in the proportions of 60%, 5%, and 35%, respectively. The training data were used by the candidate services to build their recognition models; competition data were the test cases; and evaluation data were used for evaluating the competition results.

- Experiment 1 : training and competition data belong to the same data set (*ORL*); the client submits five test cases where each test case contains two known facial images.
- Experiment 2 : training and competition data belong to the same data set (*Yale*); the client submits five test cases where each test case contains five known facial images.
- Experiment 3 : training data belong to the *ORL* data set while the competition data belong to the *Yale* data set; the client submits five test cases where each test case contains three facial images.

In each experiment, the client submitted the WSDL document of the candidate services, the corresponding test cases, and the defined competition policies to the competition desk. Figure 7 shows two instances of the corresponding test cases. Table 1 represents the competition results and the winner in each experiment (Rank =1) where we used a 2.4 GHZ dual-core CPU and a high-speed bandwidth (1 Mbyte/Sec) link between the servers and the competition desk.

Table 1: Experimental results for the data service competition using five different face recognition web services.

No	Competition Factor	PCA WS	ICA WS	LDA WS	SVM WS	NN WS
1	*Accuracy*	60%	80%	40%	80%	40%
	Confidence	38%	66%	16%	61%	12%
	Time (msec)	1690	2404	1012	6214	2605
	Score	***0.53***	***0.67***	***0.44***	***0.61***	***0.31***
	Rank	***3***	***1***	***4***	***2***	***5***
2	*Accuracy*	80%	100%	100%	100%	60%
	Confidence	49%	69%	87%	71%	39%
	Time (msec)	2237	2873	1148	8519	3012
	Score	***0.64***	***0.78***	***0.96***	***0.73***	***0.48***
	Rank	***4***	***2***	***1***	***3***	***5***
3	*Accuracy*	60%	80%	60%	80%	60%
	Confidence	43%	68%	41%	48%	29%
	Time (msec)	2023	2603	1129	7147	2078
	Score	***0.54***	***0.69***	***0.63***	***0.57***	***0.48***
	Rank	***4***	***1***	***2***	***3***	***5***

To verify whether the service competition guides the client to select the best service, we calculated the performance of each web service in each experiment. The evaluation data (50 facial images in each experiment) were selected from the same data set as the competition data. Table 2 represents the results of this evaluation where the competition desk recommended the best service (Rank=1) in all three cases. Moreover, the total ranking obtained from the competition has 73 percent accuracy in comparing the rankings on the evaluation data.

Table 2: Evaluation results for the data service competition.

No	Competition Factor	PCA WS	ICA WS	LDA WS	SVM WS	NN WS
1	*Accuracy*	57%	72%	42%	68%	33%
	Confidence	31%	39%	19%	38%	12%
	Time (msec)	1720	2411	1123	6094	2889
	Score	***0.51***	***0.57***	***0.46***	***0.49***	***0.27***
	Rank	***2***	***1***	***4***	***3***	***5***
2	*Accuracy*	76%	89%	92%	88%	64%
	Confidence	39%	61%	79%	64%	37%
	Time (msec)	2325	3103	1059	8618	2097
	Score	***0.59***	***0.69***	***0.89***	***0.65***	***0.53***
	Rank	***4***	***2***	***1***	***3***	***5***
3	*Accuracy*	52%	77%	57%	69%	51%
	Confidence	26%	59%	31%	47%	26%
	Time (msec)	2051	2526	1145	6985	2106
	Score	***0.44***	***0.65***	***0.57***	***0.51***	***0.45***
	Rank	***5***	***1***	***2***	***3***	***4***

4.2 Task Service Competition

A skin detector service [7] is a primary need in many fields including face detection, semantic filtering of web contents, video surveillance, and human motion detection. A skin detector data service requires transferring images and videos from clients to the server, which is not efficient. On the other hand, a skin detector task service which uses a client-side service representative to evaluate the skin regions offers higher performance.

Candidate Services. Several methods for discriminating between skin and non-skin regions have been proposed. They can be categorized into pixel based and block based approaches. The former provides faster results while the latter provides more accuracy. Therefore, a client should choose a skin detector service based on its performance on the specific client application.

In this case study, the client wants to evaluate the performance of three skin detector web services: pixel-based; block-based using color features; and block-based using texture features (these techniques are introduced in [17]). To show a task service outperforms a data service for skin detection, we developed both data and task services for each of the candidate services. Then, we compared their service response time for a sample image (450 Kbytes). The comparison results are shown in Table 3. Processing time and network time are the factors that influence the response time. To boost the data service performance, we considered a dedicated web service (it has no other tasks or clients) whose processor was two times faster than the client processor (i.e., 2.4 GHZ dual-core CPU). Moreover, the service client and servers were connected by a 1 Mbyte/Sec link. While a task service receives a request message (1 Kbyte) and sends a task message (up to 3 Kbytes) to a client-side service representative, a data service receives an original image (450 Kbytes) and replies by sending the modified image (120 Kbytes). Moreover, while a data service must be called for each video frame, a task service which needs to be called once assigns a task to a service representative to process the frames locally.

Table 3: Service time comparison between data and task skin detector web services.

Service Time (msec)	Pixel WS	Block-Color WS	Block-Texture WS
Task Service	582	1241	1351
Data Service	1094	1519	1594

The skin detector task services work as follows.

1. *Pixel-based skin detector*: assigns a task to the generic service representative to apply explicit rules for the color values (i.e., Red (R), Green (G), and Blue (B)) of each pixel. Different components of this task service are shown in Figure 8.
2. *Block-based skin detector using color features*: sends the following task definition as well as skin patterns (the server-side task data) to the generic service representative to customize it for this service. i) Subdivide the image into equal sized blocks; ii) extract three color features (color mean, color variance, color skewness) from each block; iii) compare each block features to all the skin patterns received from the service and choose the blocks whose Mean Square Error (MSE) is less than a defined

threshold as skin blocks; and iv) paint skin blocks black and non-skin blocks white. Figure 9 shows the task components for this web service.

3. *Block-based skin detector using texture features*: assigns a task to the generic service representative similarly to the previous service except that it considers texture features calculated by wavelet transforms for each block.

As mentioned before, a service representative is generic and can represent any services; however, the SR internal knowledge base enables the service representative to store the required business rules and actions for a specific domain. In this case study, each task service asks the competition desk to assign it a service representative who has basic image processing functions stored in its internal knowledge base. Otherwise, the service providers are asked to send the required knowledge to the generic SRs.

Test Cases. The client provides test cases in the form of (*image* , *expected result*) where the expected result is an image with the same number of pixels, but skin and non-skin pixels are assigned black and white colors, respectively.

Competition Policies. The client specifies the competition factors as *Accuracy*, and response *Time* with the following evaluation functions where FP and TN represent the False Positive (number of non-skin pixels considered as skin pixels) and True Negative (number of not recognized skin-pixels), respectively.

$$Accuracy = Average(\frac{\mid Pixels \mid - 2*TN - FP}{\mid Pixels \mid})$$
$$Time = Average(\ Response\ Time\ )$$

In this case study, the client searches for a web service that captures most of the skin pixels, therefore the accuracy evaluation function assigns more weight to TN. We used different ranking policies for each experiment (discussed next).

Experimental Results. We collected 60 random images where all images were in color with various visual qualities, details and different light conditions. These images were equally divided into competition data and evaluation data. We ran the following two experiments with our prototype system to discover the best services in each client application.

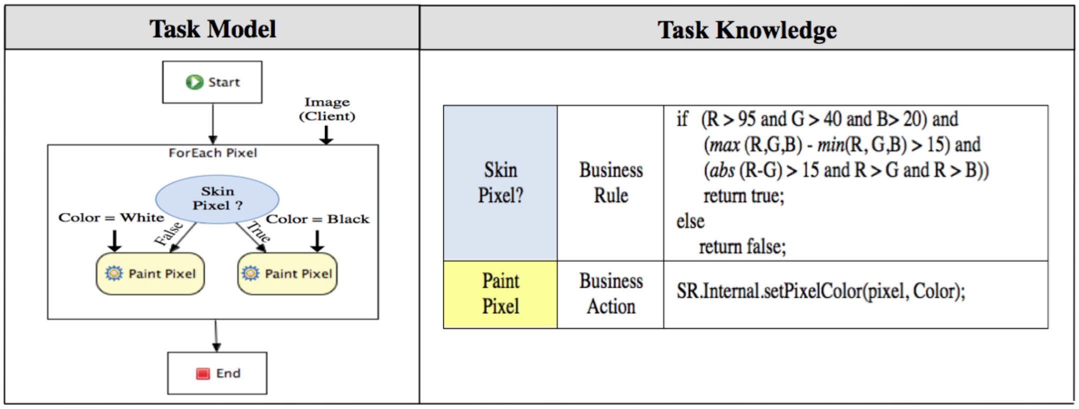
- Experiment 1: the client uses this web service for face detection; therefore the test cases contain facial images. Since the client prefers accuracy rather than speed, 0.9 and 0.1were assigned as the weights for *Accuracy* and *Time* respectively in the ranking policy.
- Experiment 2: the client uses this web service for web content filtering; therefore test cases contain body and non-body images. Since this web service will be used as a part of a live video streaming system, it needs to be fast. Therefore the client assigns 0.8 and 0.2 as the weights for *Accuracy* and *Time* in the ranking policy.

In each experiment, the client submitted five test cases (two instances are shown in figure 10). Table 4 represents the competition results and the winner service in each experiment. Similarly to the first case study, we calculated the performance of each web service on the corresponding evaluation data (which includes 30 images) in each experiment. Table 5 represents the results of this evaluation, confirming the service competition results.

5 Discussion

The proposed web service competition is differentiated from other web service selection methods as it ranks services based on their performance on a specific client's context. In the discussion section, we list a few important issues such as the applications and challenges of the proposed WS-Competition approach.

1. The proposed web service competition does not intend to replace the traditional web service selection approaches. However, this approach provides more accurate selection mechanism over the traditional approaches in the



Skin Pixel?	Business Rule	if (R > 95 and G > 40 and B> 20) and
		(*max* (R,G,B) - *min*(R, G,B) > 15) and
		(*abs* (R-G) > 15 and R > G and R > B))
		return true;
		else
		return false;

Paint Pixel	Business Action	SR.Internal.setPixelColor(pixel, Color);

Figure 8: Task service components of the Pixel WS where there is no server-side task data. In the task model, a business rule is assigned the '**?**' symbol to differentiate from a business action. Moreover, SR.Internal.x refers to the business actions stored in the SR internal knowledge base.

Table 4: Experimental results for the task service competition using three skin detection services.

No	Competition Factor	Pixel WS	Block-Color WS	Block-Texture WS
1	*Accuracy*	91%	85%	89%
	Time (msec)	829	2071	2979
	Score	***0.92***	***0.79***	***0.82***
	Rank	***1***	***3***	***2***
2	*Accuracy*	67%	83%	86%
	Time (msec)	837	2004	3709
	Score	***0.73***	***0.76***	***0.74***
	Rank	***3***	***1***	***2***

Table 5: Evaluation results for the task service competition.

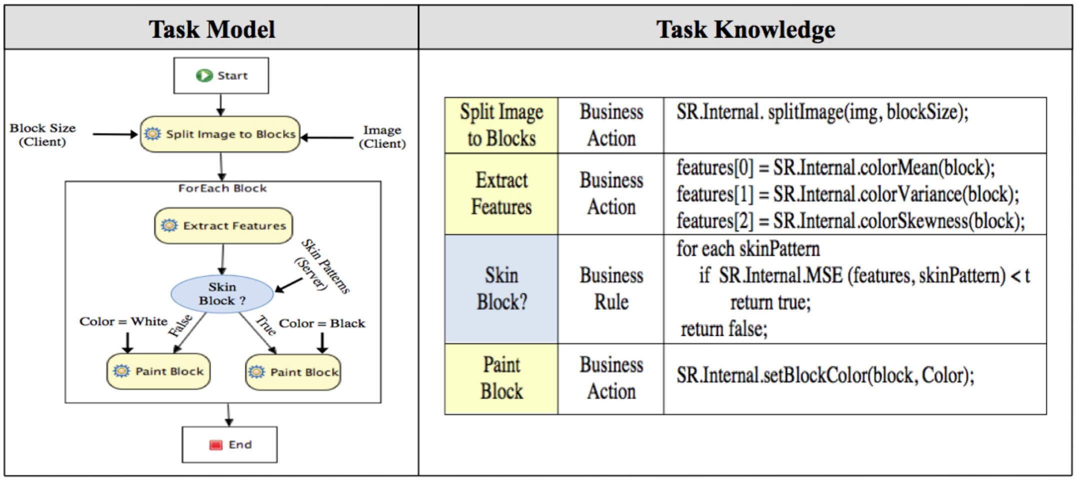
No	Competition Factor	Pixel WS	Block-Color WS	Block-Texture WS
1	*Accuracy*	88%	81%	83%
	Time (msec)	823	2012	3020
	Score	***0.89***	***0.77***	***0.77***
	Rank	***1***	***2***	***2***
2	*Accuracy*	64%	85%	81%
	Time (msec)	817	2051	3631
	Score	***0.71***	***0.76***	***0.69***
	Rank	***2***	***1***	***3***

following cases: I) The service descriptions are not accurate or comprehensive enough where the services should be ranked based on their performance. II) The service client wants to choose the best service among multiple candidate services where they represent different performance levels based on the client's context or specific application. III) The service client intends to specify the service ranking criteria to test and compare available services.

2. Service competition can not be offered free of charge for the pay-by-use services because of costs associated with tests of the competition desk. However, some services provide free trials or test versions of their web services which can be used by the competition desk. Otherwise, the competition by itself may have costs which must be paid by the service client. To support these cases, a billing model for the competition desk is required.

3. The competition desk can be provided as a web service search engine where the service client sends a query request including the category of services as well as the search criteria (competition factors). Moreover, the service client either submits its own test cases or asks the search engine to use its predefined test cases for the specified service category to hold the web service competition. Finally, the web service engine returns a ranked list of services to be selected by the service client.

4. The introduction of an actor in the SOA model and the associated tests may increase


Task Model

Task Knowledge

Split Image to Blocks	Business Action	SR.Internal. splitImage(img, blockSize);

Extract Features	Business Action	features[0] = SR.Internal.colorMean(block);
		features[1] = SR.Internal.colorVariance(block);
		features[2] = SR.Internal.colorSkewness(block);

Skin Block?	Business Rule	for each skinPattern
		if SR.Internal.MSE (features, skinPattern) < t
		return true;
		return false;

Paint Block	Business Action	SR.Internal.setBlockColor(block, Color);

Figure 9: Task service components of the Block-Color WS where the service provider sends skin patterns as server-side task data.

the required time for service selection. In most of the cases, service selection includes a long-term agreement between the service client and the service provider. Consequently, the service client prefers service selection approaches which offer more accurate and customized results over those generating their results faster. However, it may limit the applications of web service competition for dynamic service discovery which requires run-time re-binding to new and un-known services whenever the actual QoS deviates from initial estimates, or when a service is not available.

6 Related Work

Current web service discovery is based on *UDDI* (Universal Description, Discovery, and Integration), a standard for centralized repositories. Several protocols (such as *Jini*, *UPnP*, and *Salutation*)[5], middle-wares, and frameworks address service discovery, all based on the service descriptions published in a (*UDDI*) service registry. Semantic web technology has been applied to create and analyze web service descriptors. For example, *WS-Inspection* [1] is a web service specification for discovery documents that describes services at different levels and from various perspectives or [10] that proposes semantic and ontology-based service descriptors to be used in service discovery. Considering service descriptions could be promising in theory, but unfortunately most of the descriptions available are low quality. Therefore, a major challenge that web service technology faces is the discovery and selection of services, based on their capabilities and performance. That is addressed in this paper.

The integration of software agents and web services has been proposed to model the business aspects of enterprise systems, where each role or major function of an enterprise system is considered as an agent. Moreover, autonomous computing and software agents have had increasing application in service discovery. For example [8] proposes agents that rank services by analyzing the client's behaviour and satisfaction about services. The Service Location Protocol (SLP) [3] has also been used in service discovery including user, service, and directory agents. In this paper, we have assigned a new role (i.e., service representative at the competition desk) to software agents to model enterprise agents in the business domain. As an alternative to our generic service representatives, one could consider sending customized mobile agents from the candidate services to the competition desk. However, there are several security and privacy issues to be considered in mobile agent computing [4]. Mobile agent architectures also suffer from low efficiency, as they need

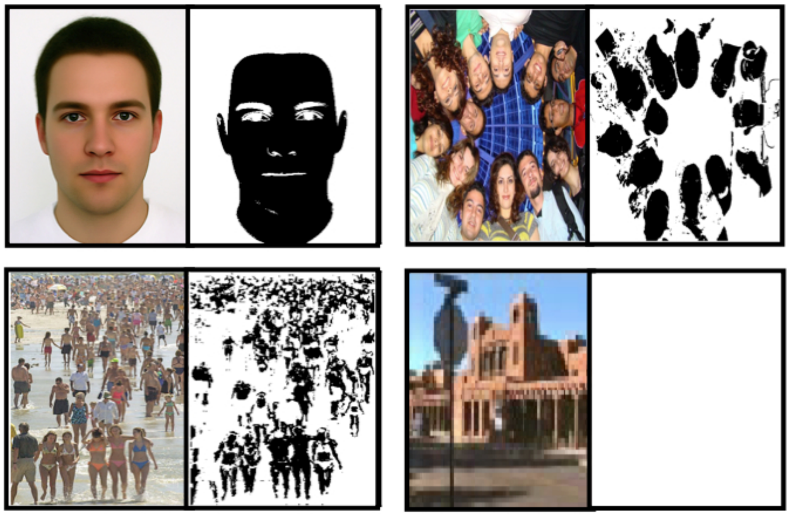


Figure 10: Four test cases *(image, expected result)* for the task service competition: (top) represents two test cases for experiment one where the client needs this service for a face detection application; (bottom) represents two test cases for experiment two where the client will use the service for web content filtering.

to transmit the entire computer program or process through the network.

As a part of the proposed web service competition, the competition desk implicitly tests the candidate web services based on the submitted test cases by a service client. Several approaches have been proposed for the topic of web service testing, as surveyed in [2]. For example, a scenario-based testing method was proposed in [18] which requires an extension to WSDL service description. Moreover, there are several approaches which propose to extend the UDDI registries with testing features. Based on these approaches, the service provider releases the test suites together with the service description to the service registries. As a result, the service registry moves toward an active role as a service tester. The proposed web service competition is differentiated from other service testing approaches by at least two factors: (i) to the best of our knowledge, the web service competition is the very first attempt to test web services based on the client's context; and (ii) the testing of client-side web services (e.g., task services) was not covered in previous work.

7 Conclusions and Future Work

For efficient service discovery, a service client needs to know both the functionality of each candidate service and the capability to perform it well for the client's specific application. Because service descriptors can not support efficiently both the functional and non-functional aspects of a service, a service client is forced to either examine many potential services or simply make a random choice. In this paper, we propose a mechanism to help a service client to select web services based on their performance and the client's needs. For future work, we plan to examine this approach in more case studies such as the competition among decision support services. The competition policies will be standardized and extended to support both functional and non-functional service features. And finally by addressing the potential interoperability issues and developing the required protocols, our ultimate goal is to extend our work to introduce a web service engine that finds and ranks services based on the proposed web service competition process.

References

[1] K Ballinger, P Brittenham, A Malhotra, W Nagy, and S Pharies. *Web Services Inspection Language (WS-Inspection) 1.0*. IBM, 2001.

[2] G Canfora and M Di Penta. Service-oriented architectures testing: A survey. In Andrea De Lucia and Filomena Ferrucci, editors, *Software Engineering*, volume 5413 of *Lecture Notes in Computer Science*, pages 78–105. Springer Berlin / Heidelberg, 2009.

[3] E Guttman. Service location protocol: Automatic discovery of ip network services. *IEEE Internet Computing*, 3(4):71–80, 1999.

[4] M Hadzic, P Wongthongtham, T Dillo, E Chang, M Hadzic, P Wongthongtham, T Dillon, and E Chang. Introduction to multi-agent systems. In *Ontology-Based Multi-Agent Systems*, volume 219 of *Studies in Computational Intelligence*, pages 15–35. Springer Berlin / Heidelberg, 2009.

[5] S Hagemann, C Letz, and G Vossen. Web service discovery - reality check 2.0. *Next Generation Web Services Practices, International Conference on*, 0:113–118, 2007.

[6] R Jafri and H Arabnia. A survey of face recognition techniques. *Journal of Information Processing Systems*, 5(2):41–68, 2009.

[7] P Kakumanu, S Makrogiannis, and N Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007.

[8] N Kokash. Web service discovery with implicit qos filtering. In *In Proceedings of the IBM PhD student symposium, in conjunction with ICSOC 2005*, pages 61–28666, Netherlands, 2005.

[9] D Krafzig, K Banke, and D Slama. *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice-Hall, 2005.

[10] U Kuster, H Lausen, and B Konig-Ries. Evaluation of semantic service discovery—a survey and directions for future research. *Emerging Web Services Technology, Volume II*, pages 41–58, 2008.

[11] M Najafi and K Sartipi. A Framework for Context-Aware Services Using Service Customizer. In *The IEEE International Conference On Advanced Communication Technology*, volume 2, pages 1339–1344, Phoenix Park, Korea, 2010.

[12] M Najafi and K Sartipi. Client-side Service Composition Using Generic Service Representatives. In *CASCON 2010: Proceedings of the 2010 conference of the Center for Advanced Studies on Collaborative research*, pages 238–252, Toronto, Canada, 2010.

[13] M Najafi and K Sartipi. Modeling service representatives in enterprise systems using generic agents. *Service Oriented Computing and Applications*, 5:245–264, 2011.

[14] J Ng, M Chignell, J Cordy, and Y Yesha. Overview of the smart internet. In *The Smart Internet*, volume 6400 of *Lecture Notes in Computer Science*, pages 49–56. Springer Berlin / Heidelberg, 2010.

[15] S Raspl. Pmml version 3.0 - overview and status. In *The ACM Workshop on Data Mining Standards, Services and Platforms*, pages 18–22, Philadelphia, USA, 2004.

[16] G Reese. *Cloud Application Architectures: Building Applications and Infrastructure in the Cloud*. O'Reilly Media, Inc., 2009.

[17] H Sajedi, M Najafi, and S Kasaei. A boosted skin detection method based on pixel and block information. pages 146 – 151, sep. 2007.

[18] A Tarhini, H Fouchal, and N Mansour. A simple approach for testing web service based applications. In *Innovative Internet Community Systems*, volume 3908 of *Lecture Notes in Computer Science*, pages 134–146. Springer Berlin / Heidelberg, 2006.