Towards Service Discovery and Subscription based on Community-of-Interest

Xuanzhe Liu, Gang Huang*, Hong Mei
School of Electronics Engineering and Computer Science, Peking University, Beijing, China, 100871
liuxzh@sei.pku.edu.cn, huanggang@sei.pku.edu.cn, meih@pku.edu.cn

Abstract

Current service discovery and subscription are “provider-centric” indeed, which costs the consumers too many efforts and makes the wide spectrum of QoS hard to use. In this position paper, we propose a “consumer-centric” service discovery and subscription approach. The core idea is derived from the “Community-of-Interest (CoI)” concept, which means the services with same or similar functionalities can form a community of the consumer’s interest. This approach facilitates the consumers in three ways. Firstly, it simplifies the service discovery for the consumers who can “purely” care about the functionality and qualities instead of the providers. Secondly, it optimizes the service subscription for the consumers who just subscribe the CoI instead of a single and monolithic service provider while the COI takes charge of matching and scheduling the “best-of-breed” service automatically. Thirdly, it incarnates the wide spectrum of QoS to the consumers who will face with all possible QoS supported by the services in the CoI. The supporting framework is discussed and some key technical issues are demonstrated.

1. Introduction

Service Oriented Architecture has emerged as a new paradigm that utilizes services as the basic constructs to support the development, deployment and management of rapid, low-barrier, and easy integration of distributed applications even in heterogeneous environment [1]. SOA allows the services can be discovered, matched, composed, subscribed, verified and executed dynamically. In a typical SOA, service providers register their services in a service broker (e.g. UDDI [13]) and publish the service description (e.g., WSDL[12]); the service consumers search and discover the required services upon the broker, and, once find, they can bind and invoke the target service in a standard way (e.g. SOAP[11]). Even more, the services from various providers can be dynamically composed as a workflow for a specific business task, such as BPEL [14] or OWL-S [15]. Obviously, SOA simplifies the application development, accelerates the system integration, and adapts various requirements in an “on-demand” way [2].

For competition pressure and time-to-market requirements, different service providers independently develop and publish their own services within an organization or on the web in a parallel way. The ever growing number of services with similar or even the same functionalities makes the service consumers have thousands of choices. In conventional SOA, the consumers must discover the services through registry (such as keyword search or category-browse on UDDI servers), and the matching may take several steps and cost a lot of time. Furthermore, too many choices may confuse service consumers to select “best-of-breed” services. In current service discovery and subscription, the consumers do find the providers instead of the services. For example, though different providers publish the services that have the same functionality, the consumers have to treat the services as different ones only because they belong to different providers. If the consumers only care about the functionality, they have to face with the providers and investigate whether different providers make the service or functionality different or not. If the consumers also care about the quality, they have to understand what qualities are implied by the providers, which is another shortcoming of current service discovery and subscription. The QoS varies in different providers and the QoS a single provider can offer is limited within a very small range. Since the consumers can only bind and subscribe a single service provider once, they cannot benefit from the wide spectrum of QoS easily and directly (although the consumers can bind, unbind and rebind multiple service providers dynamically for getting different QoS).

To our point of view, the cause of the above problems is that current service discovery and subscription is nearly “provider-centric”. In the three-party (provider, broker and consumer) model, the providers just publish the services to the brokers and wait for the consumer binding requests, the brokers just organize the services in terms of the provider’s preferences and wait for the consumer discovering requests, and the consumers have to do all remaining work in the service discovery and subscription. Considering the customer orientation promised by SOA, the consumers pay too many efforts in current service discovery and subscription.

* Corresponding author.
In this position paper, we propose a Community-of-Interest (CoI) [10] based approach to reducing the above burden of the consumers in the service discovery and subscription. For a given consumer, the services with same or similar functionalities are organized together by ontology rules and other constraints and then form a community of the consumer’s interest. The CoI acts as a unique “service entity” visible to the consumers who need not care about the providers any more. It can find the member service that satisfies the consumers most. It can also delegate the subscription process and return the results to the consumers. As there are a large number of member services with different quality levels in the CoI, the QoS spectrum becomes much larger than individual services. The CoI can be predefined in terms of the prediction or survey of the consumers’ interests or dynamically constructed in terms of the consumers’ discovery and subscription. No matter how a CoI is constructed, a consumer can put much less efforts but get a service that provides the desired functionality in a wide spectrum of QoS. In that sense, the service discovery and subscription shifts from “provider-centric” to “consumer-centric”.

The following of the paper is organized as follows: section 2 analyzes the technical issues of current service discovery and subscription and reveals the necessity for introducing CoI; section 3 analyzes the key rationales for CoI approach; section 4 proposes a framework to support the CoI in current SOA; section 5 discusses the experiments and illustrates the quality spectrum in CoI; section 6 compares some related work and section 7 ends the paper with conclusion and future work.

2. Approach Overview

2.1 Challenging Issues

The core concepts of SOA are the service registry, service provider and service consumer. Typically, the service provider publishes the service on the service broker. Once the consumer plans to discover the desired service, he/she will first send search request to the broker, discover the candidate services, choose the proper one, and finally do binding and subscription.

Take web services as the supporting technology for current service discovery and subscription. The discovery on UDDI is organized as a tree-like search [13]. The provider should first register as a “BusinessEntity” on the UDDI server, and publish the services in “BusinessService”, which are the leaf nodes. Once consumers plan to request the service, they must view the service description via UDDI with necessary steps, by keyword search, category browse, or mixing both. If the search reaches the service that meets the consumer, he can bind and subscribe it through “BindingTemplate”. This procedure is illustrated in Figure 1.

![Figure 1 Traditional Discovery and Subscription](image)

This mechanism is really “provider-centric”, because the providers determine what services can be discovered and subscribed. It makes sense when the number of candidate services is relatively small. However, the very nature of SOA is that thousands of providers can publish their own services which may have very similar or even the same functionalities but different QoS. Though current UDDI supports searching by the white page, yellow page and green page, the consumers would still have so many optional but puzzling choices that they will pay much time and many steps to view these services. With the ever growing number of services, it will become more difficult to the consumer to do discovery task. Moreover, current service subscription is typical “End-to-End”, which means only a single provider can be bound and subscribed by the consumer. Usually, a single provider can satisfy the functional requirements of a consumer but cannot provide a wide spectrum of QoS. On the other hand, the function-similar services can aggregate their QoS and their discovery, selection and subscription become an important and challenging issue in the area of SOA research [1], which cannot be handled by the “end-to-end” subscription.

2.2 Community of Interest

The difficulty of service discovery and subscription is well-known but two important facts may give us elicitation: Firstly, it is true that a number of services are provided by different vendors, but the differences are mainly about non-function properties. Most of the functionalities are very similar or even the same, only vary from the operation name or parameter type. Secondly, for a consumer’s lookup request, there might be a set of services returned. If anyone of the set can satisfy the consumers, the consumers do not need to browse the returned services one by one. Therefore, can we aggregate these function-similar services into a single
“virtual community”? This community serves for the specific requests, e.g., Travel Agency. For all consumers, they only interact with the community service instead of individual services. The community acts as the portfolio and manages the real services, receives the consumer’s requests, dispatches them to the proper target service provider, and coordinates the task scheduling between services.

Obviously such idea is based on the consumer’s interests of some specific functionalities, and simplifies the consumer’s task while also enlarges service quality spectrum. By our investigation, it is very like the Community-of-Interest (CoI) from the knowledge management domain [9]. The CoI refers to the groups of people (typically coming from different disciplines) which engage in a joint activity. It is the collection of people that are concerned with the exchange of information in some subject area [10].

The initial purpose of CoI is to make knowledge share between users with similar interests [10]. To our point of view, the CoI can be naturally applied to SOA, that is, the CoI in SOA is the collection of several member services with the similar or even the same functionalities. In other words, the member services within a specific business domain are organized together as a unique “community” to offer different quality-of-services (QoS) for one or a set of given consumer requests.

The CoI evolves current service discovery and subscription to “consumer-centric”. The conventional service registry, such as UDDI, is organized in tree-like categories [13]. In addition to the strict tree-like classification, the CoI provides a very useful approach to classifying the services according to not only the functionalities but also the business domain and customer interest. Thus, the CoI simplifies for the convenience of the automatic service identification, discovery and matching. For example, a Travel CoI may contain various services associated with travel so that both consumers and providers of travel can have easy access. At the consumers’ side, once they submit their requests, they do not need to face the fact that thousands of candidate services are to be selected. They only interact with the CoI service instead of browsing the member services one by one. On the other hand, as the CoI acts a legal and trustworthy service community, the consumers need neither to consider the trust nor security for target services. They sign a service contract with the CoI only. The CoI takes charge of parsing and dispatching the request, and finding the “best-of-breed” service registered in the community. In this way, the CoI can provide different consumers with different service quality levels in a consumer-transparent way. Moreover, the CoI is even a mechanism that supports high dependability: the requests to a failed or tentative unavailable member service can be automatically dispatched to another one, or the CoI provides some other exception handling actions to ensure the transactional properties.

The CoI also shows a new perspective for service management. It is obvious that the CoI is a way that aggregates independent member services all over the internet to improve service resource utilization. Once the providers publish their services to a CoI and the services are involved after verification and validation, they do not need to interact with consumers anymore since the CoI coordinates the consumer requests to proper member services and delegates the task scheduling between member services. So, all the services are well organized and managed by the CoI, the negative competition and fraud providers can be controlled.

3. Key Rationales

3.1 Not Break, But Enhancement

As discussed above, the CoI really provides a new approach for service discovery and subscription. However, it should be noted that the CoI does not break up but enhances the current SOA paradigm. From out point of view, the CoI provides some new features in addition to the three-party model, as shown in Figure 3.
- **Service Provider**: the service provider still does the service publish task. Compared to conventional service registration, they need to provide some additional information besides functional description only, such as some QoS properties or the usage scenario [4], etc.
- **Service Broker**: the service broker takes charge of the CoI construction. In our opinion, the construction could be both pre-built and on demand. The broker should also provide the service verification, validation and function testing.
- **Service Consumer**: the discovered and subscribed objects for the consumer are no longer the services, but the CoI. The CoI schedules the requests to target services, and delegates the binding task for the consumers in a transparent way.

![Figure 3 SOA enhanced by CoI](image)

Therefore, the CoI takes up the core position in the three-party model. As a CoI is setup according to consumer-specific requests, it improves the SOA to be more “consumer-centric”. Enabling the CoI based SOA requires: 1) the build-up mechanisms to construct a CoI; 2) the development of efficient data structures, mechanisms and protocols to support service discovery and access in a consumer-transparent way; 3) the policy of a CoI to govern the member services and their collaboration.

### 3.2. CoI Construction

The CoI construction is the first and the most vital step in the approach. There are two different ways:
- **Pre-built**: it is a rather static way. Once the services have been registered, they will be organized to a CoI. That is to say, the participation to a CoI is done before the subscription. So, when the consumers submit the request, they can discover existing CoIs very fast. The pre-built CoI is rather long lived and often satisfies those consumers specific to a domain.
- **On Demand**: compared to the pre-built way, the CoI constructed on demand is dynamic and temporary. Here the CoI is setup after the consumers submit their requests. Such way is exactly for more personalized requests, and the CoI only lives during the subscription. Once subscription is finished, the CoI dismisses as well.

Obviously these two construction ways provide consumers with enough flexibility. The pre-built one is faster, while the on demand one is more personalized. Both ways require three key supporting technologies:
- **Ontology Foundation**: It describes the service meta-data, standard terminology and reference models. The ontology is organized dictionaries of service, including the service parameters, interfaces and messages information, service domain knowledge, usage description. It also includes the Service Level Agreement description (QoS) and service metering, etc [2]. The ontology foundation determines which services can be taken into the same CoI.
- **Conformance Assurance**: the services in the same CoI should be guaranteed the conformance from three levels: service functionalities, behavior and semantics. Functionalities conformance ensures the CoI can serve a specific task or business domain. Behavior conformance ensures the correctness of logical relationships between the member services if we need to composite them for high-level operations. For example, once a service fails to process a request, the CoI may dispatch it to another one as a substitution. The behavioral conformance makes the state replication and compensation be achieved in a correct and unambiguous manner. The semantic conformance ensures the services are annotated with domain-specific or scenario-specific ontology properties so that they preserve their meaning when they are formally verified and validated.
- **Function Testing**: as the CoI will be a very strict and legal service entity to serve consumers, its trustworthiness is of great importance [3]. The service can not be adopted into a CoI until it passes the verification and validation successfully. The providers should make sure their services can correctly serve the consumers. Once a service provider registers its service, the CoI orders it must provide its testing scripts as well. The CoI executes the testing scripts, once passes, the candidate service is registered as a member into a CoI.

### 3.3 CoI based Discovery and Subscription

The major benefit the CoI brings is simplifying the discovery and subscription for the consumers. These two issues are supported as follows:
- **Discovery and matching**: the CoI acts as a delegate that takes charge of discovery and matching according to consumer’s requirements. Compared to traditional keyword search or category browse, the CoI allows consumers to submit their requests as a
template. The CoI parses the templates, and makes discovery and matching from two levels, including the function level that exactly matches in the parameter types, interface specifications and messages and the semantic level that retrieves the service profile, such as QoS and usage descriptions. It should be noted that the discovery and matching are completely processed transparently for the consumers, that is, what they interact with is the CoI only, not the real target services.

- Dynamic subscription: the service subscription in a CoI is unlike traditional SOA, i.e., the consumers do not interact with the real target service, instead, they only subscribe the CoI who routes the request and returns the results to the consumers. From a logical view, a CoI is a multi-tenancy service that can be subscribed by different consumers simultaneously. Thus, one important issue here is to generate a “unified” interface to all consumers and make well-defined mapping to the real target services. This also requires the CoI can dynamically dispatch the requests in “on demand” way based on changing workloads, ensuring SLA, reliable messaging delivery [7] and security.

3.4 CoI Management

A CoI is a “logical” but not a “physical” community over the services across the internet. The management of a CoI needs to be dynamic and flexible.

- Monitoring and Management: the CoI is just a virtual community hosting the real services, and the services are physically hosted by their providers. For various reasons, the services may change from time to time. The changes may be at the quality level, such as the cost or response time, or at the functionality level, such as the usage change or version update, or even at the availability level, such as the crash of service or network jam [8]. So there must be powerful monitoring and management support to ensure the CoI can work. The CoI should monitor the events and information produced by the services. We can apply agents or reflection to do the monitoring just in time. The CoI management mainly refers to the validation of the member services. With the data and information from monitoring, the CoI should verify and validate those changed services, determine whether they can remain in the community or not, and update the member service list.

- Policy enforcement: the services capabilities and requirements can be expressed in terms of policies [1]. In our opinion, the policy can be applied in the CoI from two perspectives. One is to manage the community itself, for example, to establish some rules to constrain the eligibility to the member services based on ontology. Another is to organize the interaction and relationship between the member services. For example, as we discussed before, if one service fails, the CoI can dispatch the request to another one as substitution. Here, the state replication, synchronization and compensation should be defined by policies. Obviously, there should be a CoI portfolio who takes charge of the policy.

4. Supporting Framework

We propose a framework in Figure 4, which provides support to the key rationales above. In the following section, we will illustrate the components and how the framework works.

![Figure 4 Approach Framework](image)

- Service Broker: It is the repository where stores the service profile, including the interface specification, contract and SLA information. In addition to current UDDI server, the broker also stores the ontology repository and testing scripts.

- Management: it provides two major management capabilities, registration management and verification & validation. All services to be registered to broker should firstly pass the verification & validation, and match the ontology repository. This needs the semantic similarity match [5]. Once passed, they will be adopted to a CoI. It should be noted that the management is on-the-fly, for it receives the information retrieved by monitoring, and makes update or removal of services that has been registered.

- Monitoring: it dispatches an agent to every registered service, to monitor their status and change. The monitoring is even driven, and the agent retrieves and collects the service information. Once the service is changed, updated, removed or temporary unavailable, the retrieved data will be fed back to the monitoring component, and sent to the management components.
Policy Control: the policy control component takes charge of establishing the community rules to define a community, and managing the relationships between the member services.

Selection Engine: it is the major component to process discovery and matching. It consists of three components. The Planner parses the consumer’s requests, queries the broker and returns the details of each member service. It can generate the selection policy automatically with the given constraints. It should be noted that the planner can work just-in-time and multi-tenancy, which fits different consumers simultaneously. The execution component is the core component that retrieves the selection policy and selects the services at runtime. It generates a “service pool” which can all meet a specific request, and determines which member services will be selected for or removed from the pool. The service pool is an available service queue for the service consumers. There are some existing mechanisms to build such pool [7]. Once the current pool is unavailable or brings a significant performance penalty (such as current service costs too much time over expected), it triggers the Exception Handling and requests the planner to replan the pool policy. The Exception Handling also processes the compensation and coordination between different member services to ensure the transactional properties, member services’ rights, and so on.

CoI Core: is the core component of CoI. It initializes and manages a CoI, and takes charge of the interaction to consumers. As discussed before, the CoI core should support both pre-built and on-demand constructions. In our framework, the former is done when services pass the registration, while the latter works only when consumer’s request arrives. To every consumer’s request, the CoI core requests the selection engine, discovers and matches the proper services (notice it is the service pool maintained by the engine). However, for the consumer, he only sees one service interface specification, which is generated by CoI core. The CoI core should also route the binding message to the real target member service for the consumers. However, as the CoI is a multi-tenancy community, CoI also maintains the correlations for different service consumers. As there are may be coordination between member services, CoI core also acts the “community portfolio” to adapt the community rules and manage the member relationships.

5. Experimental study

The CoI approach can simplify the discovery and matching, and provide the “best-of-breed” service to the consumers. For illustrating the first benefit, a new human-computer interface in service discovery is under development. For the second benefit, the selection and scheduling policy is a critical issue. We have designed a series of algorithms and experiments based on web services [7].

The algorithm focuses on the response time optimization when meeting high reliability requirements. It contains two parts: Average Response Time (ART) and MRT (Minimum Response Time). ART discovers the member services set that can meet the high reliability level within shortest response time overhead. MRT can make the consumer’s waiting time in minimum if his requirements can not be met by current CoI.

The experiments are conducted on a server and several client machines. All the machines are Pentium IV 2.8G PC with 512M DDR memory. These machines are directly connected with 100 Mbps Ethernet.

Firstly, to simplify the construction of CoI, we develop 10 services having same functions, deploy them in the Apache AXIS engine and publish them in the Apache JUDDI registry. Then we search for services, consume every service and provide feedbacks to record the response time and reliability.

Four different selection policies in the CoI here are: (1) non-optimal selection in random sequence; (2) optimal selection with the optimal algorithm; (3) reliability prior selection in reliability descending order; (4) response time prior selection in response time ascending order.

Two assumptions are held in the experiment: (1) we do not consider the binding time of request, for all the services are deployed in the same net-work environment; (2) the sort duration is neglected. We apply MergeSort, which costs less than 20 milliseconds and such time cost can be neglected compared to the execution time of a service.

First we investigate the ART-prior algorithm to illustrate the improvement of subscription. We constraint the consumers’ reliability requirement ranges from 85% to 95%. Figure 5 shows the trend for reliability with the number of services in CoI. No matter what selection policy is applied, we can see that the reliability increases when the CoI size increases. This gives us the confidence that the reliability spectrum will be enlarged by requesting a CoI. And, the ART-prior algorithm also shows the effectiveness of providing “best-of-breed” services by CoI. Besides the reliability prior selection (it always selects the service having higher reliability than others), the optimal selection can achieve a little higher reliability than the random selection and the response time prior selection. Another result we get is that with optimal algorithm, the CoI can achieve lower response time penalty for consumers, as shown in Figure 6.
Another conclusion using the optimal selection algorithm we get is that the reliability keeps stable after the CoI involves 4 instances (here the reliability is 95.4%). It reveals the fact that the first four instances are reliable enough in most cases, while the random selection needs more instances to reach the same reliability. Therefore, by the optimal policy, it is unnecessary to enlarge the capacity of CoI continuously and then the maintenance cost can be well controlled. We have more detailed discussion on it in [7].

Figure 5 reliability comparison

Figure 6 ART response time evaluation

MRT-prior algorithm to illustrate that a CoI can make consumers pay minimal cost if their request can not be met. Here we provide the consumers much higher reliability, i.e. 98%. As shown in Figure 7 it is obviously that the optimal selection always resides below other three algorithms. It reveals that optimal selection guarantees the lowest time overhead. The random selection has the worst MRT, since it may try all service instances.

The simulations above may be a little simple and only consider two QoS attributes, however, the results have demonstrated the rationality of consuming the CoI instead of the member services individually. So with certain algorithms and mechanisms supporting, the CoI can work as expected. In the future work, we will consider more QoS attributes and investigate other effective algorithms to coordinate the CoI.

Figure 7 MRT response time evaluation

6. Related Work

The concept of CoI was originally introduced in [10]. This paper introduces the foundations of CoI, such as ontology, semantic and policy control. Furthermore, we involve these concepts to make SOA “consumer-centric”, propose novel frameworks and retrieve key rationales for CoI based SOA.

The work proposed in this paper is related to several other research efforts in the area of SOA including web services and semantic web. Due to space limitation, we limit our discussion to works which support consumer-centric and the CoI construction.

The most similar work to our approach is the one proposed in [3][4]. Based on current Service-Oriented Architecture (SOA), W.T. Tsai et al proposed a collaborative software design methodology in an SOA Environment: the Global Software Enterprise (GSE). They also considered current SOA is producer-centric, in which the service providers publish services that they produce and let the consumers to search available services to compose their applications. GSE supports the application builders to publish their application requirements for the service providers to follow when producing or customizing services to support the application. Compared to their work, our approach focus on the “service resource aggregation”, and pays much attention to simplify the consumers’ task. It makes sense that these two works can cooperate, for example, the requirements templates on GSE can be reused in the CoI ontology foundations.

Initializing and constructing a CoI is the first and the very vital step in our work. It is an advanced topic, for service matching is based on semantic similarity. In[5], Xin. D etl proposed a data mining based approach for discovering similar web services at the level of finding similar operations and finding operations that compose
within a given one. In [6], Jeffrey presented a numeric metric for calculating the semantic similarity of services described by OWL-S [15]. Their similarity measure is based on the intuition that similar services share common descriptive information. This work can help us to investigate the similarity between services and improve the construction of CoI. However, the web services and web ontology are still not mature enough to support the perfect service discovery/matching.

7. Conclusion and Future Work

The CoI makes the SOA from “provider-centric” to “consumer-centric”. This paper presents the concepts, architecture, key rationales, and illustrative experiments for the CoI based service discovery and subscription. Such “consumer-centric” approach simplifies the discovery, matching and subscription task, reduces the workload, and enhances the QoS level. It also extends the capacity of enlarging the QoS spectrum for multiple consumers.

The framework and the implementation on web services are currently under development. With real life use cases, we also aim to investigate the requirements templates that CoI can provide to the consumers, with plentiful information to subscription more easily. There is another interesting topic, CoI collaboration, which tries to establish the collaborative workflow among multiple CoIs. It will even simplify current business process model.

8. Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 60233010, 90612011, 90412011, 60403030, the National Grand Fundamental Research 973 Program of China under Grant No. 2005CB321805 and the IBM University Joint Study Program.

9. References


