Abstract—The Pragmatic-context acts as an important role in pragmatic web oriented services discovery where the context is changing rapidly. This paper proposes a method for services discovery fitting to the dynamic environment called pragmatic-context. We figure out the ways include how to represent the context, how to determine it, and how to use it to influence the activation of user preferences to provide user with more tailored personalized services.

Keywords: Web;Service;Discovery;Pragmatic-Context

I. INTRODUCTION

Different to syntactic Web and semantic Web, Pragmatic Web services are aware of their current environment, i.e. services are context-aware. The shift to pragmatics has the following principles for building pragmatically sound web systems[1]: User before provider, Process before data and Interaction before representation.

Services discovery should provide users with a more personalized behavior efficiently. Z. Maamar described that personalization involves a process of gathering user information during interaction with the user, which is then used to deliver appropriate content and services, tailor made to the user's needs. The aim is to improve the user's experience of a service[2]. It is shown for instance that the needs of mobile users regarding information access are quite different from the needs of stationary users [3]. For example, mobile users' needs are not about browsing the Web, but about receiving personalized content that is highly sensitive to their immediate environment and respective requirements[4].

Context of usage is one focus of service discovery based on pragmatic Web and is all-important to deal with issues like information overload and relevance of information [5]. We figure out the ways include how to represent the context, how to determine it, and how to use it to influence the activation of user preferences.

II. PRAGMATIC-CONTEXT FOR WEB SERVICES

A. The Content of Pragmatic-Context

Z. Maamar proposed three types of context such as U-context of User context, W-context of Web service context, and R-context of Resource context[4]. We extend it more detailed for pragmatic-context including User-context, Web-context, Client-context, Resource-context and Activity-context.

User-context covers the basic information of the user (e.g. ID, name, age, sex, occupation, location ) reflects his personal preferences and current state. Location information contains data about the consumer's current location, e.g. the consumer's current address, GPS coordinates, country, or local time and time-zone. Location context may also include semantic location information, e.g. that a consumer is currently at work[6]. Web-context describes the awareness of the current state of a service and the running constraints on the service. Client-context comprises information about a client's device such as hardware and software configuration. Web-context is used for Web service adjusting its running manner to the client device’s properties. For example, Web service can choose a proper output view and data compression upon the user's network access such as 100M Ethernet, GPRS or MODEM. Resource-context records the remain proportion of the consumed resources or the busy/idle state about critical facilities and so on. Activity-context answers what is occurring in the situation and tracks the status of interaction between user and services.

B. Meta Properties of Pragmatic-Context

As machine understandable information for the web, Metadata provides the essential link between the information creator and the information user. Based on metadata, we describe and manage the objects(e.g. services, data, resources) in Web environment.

The attributes of pragmatic-context are characterized by metadata which exposes the relation among attributes such as restriction, dependency, etc. We define such metadata as meta-properties which is provided with following presentation[7]:

1) Rigidity: A property essential to all its possible instances.
2) Identity: It refers to the problem of being able to recognize individual entities in the world as being the same.
3) Unit: It refers to being able to recognize all the parts that form an individual entity.
4) Dependency: A property P is specific dependent on an other property if and only if, for all its instances, there exists instances of Q which do not overlap with instances of P, that is Q may have parts that are not shared with P.

Meta-properties for pragmatic Web services are used for description various context information including host,
media, dada, device, service, link, user, syntactic and semantic information. We classify such mess information into five types of context consist of location, identity, activity and state which are more important than others in practice.

C. Pragmatic-Context Abstract Description

Domain ontology can present precisely the content of conceptions in the domain and the relations among these conceptions. Pragmatic-context ontology\[8\] is defined as follows:

Definition 1: A domain ontology is a tupel \{C, R, Hc, rel, Ao\}, where C is a conceptions set; R is a relations set; Hc is the hierarchy of conceptions; rel is the relation between conceptions and Ao is ontology axiom\[9\].

We introduce OWL recommend by W3C as the ontology description language. Domain ontology is a normal description of domain concepts and relationships. With the help of domain ontology, we can precisely describe service interface and abstract a service as a entity with input and output properties. Thus, we denote pragmatic-context as follows:

Definition 2: Pragmatic meta-property: PMP = ( Name, Value, Range ), where Name is the name of pragmatic meta-property, Value is the value of the PMP, and Rang is the bound of the value.

Definition 3: A pragmatic-context vector can be defined as PTV = (c1, c2, …, cn), where ci (i = 1, 2, …, n) is a PMP. Thus, we present the pragmatic-context information in the service interaction procedure as (PTV1, PTV2, …, PTVm).

For example, User-context = (name, sex, age, special-demand, preference), where special-demand comprises a verb and a parameters set including target, direction, way, beneficiary and time\[10\]. Assumption special-demand = (verb, target, direction, way, beneficiary, time), thus, User-context is extent as (name, sex, age, verb, target, direction, way, beneficiary, time, preference).

Definition 4: pragmatic-context set for services: PCSS = \{ User-context, Web-Context, Client-Context, Resource-Context, Activity-context \}.

III. PRAGMATIC-CONTEXT ARCHITECTURE

A. Process Type

There are two pragmatic-context process types can be accepted: Explicit Context Processing and Automatic Context Processing. We use Automatic Context Processing for pragmatic-context layer and framework based on Markus Keidl’s following statement \[6\].

Explicit processing means that Web services or clients directly access the context contained in the service request message and the context processing functionality is part of Web Service code. Thus, there is a tight coupling between such Web services and clients and the context types they are able to process.

Automatic context processing means that service messages are pre and/or post-processed (from a Web service’s point of view) based on the context information they contain. Automatic context processing is done by the context framework, i.e., Web services are not involved in it. As a consequence, the context processing task is moved from the Web services to the service platform and the coding effort for Web services is reduced.

B. A Layer Architecture

A layer architecture for using pragmatic-context is depicted in Figure 1. It is built around a classical layered model: physical, context and Web service layers\[8\].

![Figure 1. A Layer Pragmatic Architecture](image_url)

Physical layer represents the resources belonging to the environment including all the entities that may be involved in the execution process of an application; so they describe physical devices, software components and sensors, etc.

The context layer is the important part of the architecture. It is responsible for carrying out the process of collection context and deal with it. The context layer comprises context components and pragmatic-context ontology base. Context components including aggregation, discovery, decision and explain modules and provide interfaces for upper layer. Due to the dynamic changes in the availability of physical devices, network bandwidth, connectivity and user location, the system has to classify those resources monitoring constantly their changes. In such a way the upper layers have always a consistent description of the physical dimension\[11\].

The Web service layer is the core part of service logic functionalities. It is responsible for discovery service, composition service and execution the customized and personalized services.

IV. PRAGMATIC-CONTEXT IN SERVICE COMPOSITION

A. Services Composition Procedure

Here, we classify services into two types: atom-services and compound-services. Atom-service is a service with integrity input and output data that can be no longer divided.
Compound-service consist of several atom-services integrated according to the given composition logic.

Definition 5: Service-Class(SC) is such a set consist of atom-services with the same functionalities. Elements of the set can replace each other on web functionalities with only different QoS. For services composition, each atom-service of the same set is called a candidate service of the service-class. Thus, service-class also is named a set consist of candidate services on a special task. The pragmatic web services composition model in this paper is about how to select a non-inferior candidate service according to the current context[8]

In services running procedure, the logic and time relations are decided by the requested functionality. The purpose of service composition is to select a non-inferior atom Web service(AWSi) from each service-class(SCi). Those selected AWSs together provide the user a whole service procedure named composition web service. The part of output of AWSi(Oi) can be pass to AWSj as input message(Ij). Thus, the relation between AWSi, and AWSj(i-j) obeys I/O logic constraint.

In the model, we assume composition Web service(CWS) is composed of some atom-services, otherwise we can divide those non atom-services into several atom-services[8]. Thus, the composition Web service is described as an atom-services sequence(AWS0, AWS1, …, AWSn-1) of length n.

\[
\text{Figure 2. The Running Order of Atom-Services}
\]

In figure 2, we only consider the running order relations between atom-services. We define the procedure of an atom-service running as a service time unit (STU) without considering how long it is. We adopt STU for measuring the long of compound running services procedure. The count of STU increases by step 1 whenever an atom-service process is over. Every one STU period, we check the change of pragmatic-context. Thus, the whole pragmatic-context(C) of a CWS is defined as C = (c0, c1, …, cn), where ci (i∈[0 , n], n is the count of STU) is the pragmatic-context at every STU time point, c0 is the initial state at begin of CWS and cn is the final state at the end of CWS.

Assumption T_CWS is a service-class subset selected from all services-classes upon the user’s request, T_CWS = (T_AWS1, T_AWS2, …, T_AWSk), where T_AWSi (i∈[1 , n]) is a service-class at STU time point i. Given the service-class T_AWSi comprises k atom-services, the selection target at STU time point i is to choose a AWSj (j∈[1 , k]) with non-inferior QoS from T_AWSi according to the current pragmatic-context(Ci).

B. A Service Discovery Algorithm

We define the services set at time point t (measure by STU) as Si = Fi(S, PCSSi), where S is the all services set; PCSSi is the pragmatic-context at STU time t; F is the service choosing algorithm. We choose a candidate service from the service-class at time t using decision-tree in which each node stands for a candidate service. The node with the max expected QoS value calculated by following algorithm will be chosen as the winner under the current pragmatic-context constraint[12].

The expected utility function EPi: PCSSi×Rt(ξ, pi)→[0, 100] figures out the effect on a given service QoS target ξ impacted by the pragmatic-context item pi. Where PCSSi stands for the pragmatic-context set for services at STU time t and Rt(ξ, pi)≥0 indicates the value of target ξ with parameter pi e.g. the value about time or cost[12]. Here, we assume VQoS denotes the final expected effect on the QoS impacted by all pragmatic-context items(pi1, pi2, …, pin) at STU time t together, thus, the expected QoS value of each node in decision tree is computed by follows equation:

\[
V_{QoS} = \sum_{i=1}^{n} \left[ \frac{\omega_i}{\sum_{j=1}^{n} \omega_j} \times EP_i(T_i, \xi) \right]
\] (1)

Where \(\omega_i\) is the weight of context parameter Ti upon the user preference[12].

V. CONCLUSION AND FUTURE WORK

We introduced a pragmatic-context model and explained it in detail including content, functionalities, role and framework. Our main contribution is proposing the User Context Home Server which supports getting user-context information quickly and precisely wherever the user roams. Currently, many researches about pragmatic-context focus on the usage in services composition and procedure. For the future, we plan to further investigate additional context types and to study context processing instructions in more detail. A further issue of interest is how to tune the pragmatic-context ontology for the services about multi-domain application to assure the consistence of pragmatic-context interaction procedure.

REFERENCES


