รายการอ้างอิง

[1] Kreger, H. Web services conceptual architecture (WSCA 1.0) [PDF file]. IBM Software Group, May 2001, Available from: www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf [October 13, 2006]

[2] Korkea-aho, M. Context-Aware applications survey [Online]. Internetworking Seminar (Tik-110.551), Helsinki University of Technology, Spring 2000. Available from: http://users.tkk.fi/~mkorkeaa/doc/context-aware.html [October 13, 2006]

[3] Dey, A. K., and Abowd G. D. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22. College of Computing, Georgia Institute of Technology, Atlanta GA, 1999

[4] uddi.org. UDDI [Online]. Available from: http://www.uddi.org [October 13, 2006]

[5] Gruber, T. R. A translation approach to portable ontology specifications. Knowledge Acquisition 5, 2 (June 1993): 199-220.

[6] Bearman, M. Y. ODP–Trader. International Conference on Open Distributed Processing (ICODP'93), pp. 19-33. Berlin, Germany, September 13-16, 1993.

[7] Object Management Group. Trader Object Service Specification. Revised Edition, March, 1996.

[8] w3c.org. SOAP [Online]. Available from: http://www.w3.org/TR /soap [October 13, 2006]

[9] w3c.org. WSDL [Online]. Available from: http://www.w3c.org/TR/wsdl [October 13, 2006]

[10] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43, 5-6 (November/December 1995): 907-928.

[11] Costello, R. L. and Jacobs, D. B. OWL Web Ontology Language. The MITRE Corporation, 2003

[12] Pokraev, S., Koolwaaij, J. and Wibbels, M. Extending UDDI with context-aware features based on semantic service descriptions. Proceedings of the 1st

International Conference on Web Services (ICWS'03), pp. 184-190, Las Vegas, NV, June, 2003.

[13] Lee, C. and Helal, S. Context attributes: An approach to enable context-awareness for services discovery. Proceedings of the Symposium on Application and the Intenet (SAINT), pp 22-30. FL, 2003.

[14] Mostefaoui, S. K., Gassert, H. and Hirsbrunner, B. Context meets web services: Enhancing WSDL with context-aware features. Proceedings of 1st International Workshop on Best Practices and Methodologies in Service-Oriented Architectures: Paving the Way to Web-services Success, pp. 1-14. Vancouver, British Columbia, Canada, 2004.

[15] Doulkeridis, C., Loutas, N. and Vazirgiannis, M. A system architecture for context-aware service discovery. Proceedings of International Workshop on Context for Web Services (CWS'05), pp 101 - 116. Paris, France, July 5, 2005.

[16] Doulkeridis, C. and Vazirgiannis, M. Querying and updating a context-aware service directory in mobile environments. Proceedings of the 2004 IEEE/WIC/ACM Web Intelligence Conference (WI'04), pp. 562-565. Beijing, China, September 562-565, 2004.

[17] Keidl, M. and Kemper, A. Towards context-aware adaptable web services. Proceedings of the 13th international World Wide Web conference - Alternate Track Papers & Posters, pp. 55-65. NY, 2004.

[18] ShaikhAli, A., Rana, O., Al-Ali, R. and Walker, D. UDDIe: An extended registry for web services. Proceedings of Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT 2003, pp. 85-90. Orlando, FL, January 27 - 31, 2003.

[19] Schade, A., Facciorusso, C., Field, S. and Hoffner, Y. Advanced dynamic property evaluation for CORBA-based electronic markets. Proceedings of the 2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000), pp. 109-116. Milpitas, CA, June 8-9, 2000.

[20] Sriharee, N. and Senivongse, T. Matchmaking and ranking of semantic web services using integrated service profile. International Journal of Metadata, Semantics and Ontologies 1, 2 (2006): 100-118.

[21] apache.org. jUDDI [Online]. Available from: http://www.apache.org/juddi/ [February 23, 2007]

[22] IBM Corporation. UDDI4J [Online]. Available from: http://uddi4j.sourceforge.net/ [February 23, 2007]

[23] Hewlett Packard Laboratories. Jena2 [Online]. Available from: http://jena.sourceforge.net [February 21, 2007]

[24] University of Maryland. Swoogle [Online]. Available from: http://swoogle.umbc.edu [February 23, 2007]

[25] Dun & Bradstreet. D-U-N-S [Online]. Available from: http://www.dnb.com [February 27, 2007]

[26] Universal Standard Products and Services Classification. UNSPSC [Online]. Available from: http:// http://www.unspsc.org [February 23, 2007]

ภาคผนวก

ภาคผนวก

ก. ตัวอย่างออนโทโลจีที่นำมาใช้ในการทดสอบ

ก-1. ตัวอย่างออนโทโลจีซอฟต์แวร์

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/mpeg4player.owl#"
  xml:base="http://www.owl-ontologies.com/mpeg4player.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="MediaPlayerClassic">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="MPEG4Player"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="AVIPlayer"/>
  <owl:Class rdf:ID="RealPlayer">
    <rdfs:subClassOf rdf:resource="#AVIPlayer"/>
  </owl:Class>
  <owl:Class rdf:ID="WindowsMediaPlayer">
    <rdfs:subClassOf rdf:resource="#AVIPlayer"/>
  </owl:Class>
  <owl:Class rdf:ID="QuickTime">
    <rdfs:subClassOf rdf:resource="#MPEG4Player"/>
  </owl:Class>
```

```
</rdf:RDF>


<!-- Created with Protege (with OWL Plugin 2.1, Build 284)
http://protege.stanford.edu -->
```

ก-2. ตัวอย่างออนโทโลจีอาหาร

```
<?xml version="1.0"?>

<rdf:RDF

    xmlns="http://www.owl-ontologies.com/foodcategory.owl#"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

    xmlns:owl="http://www.w3.org/2002/07/owl#"

  xml:base="http://www.owl-ontologies.com/foodcategory.owl">

  <owl:Ontology rdf:about=""/>

  <owl:Class rdf:ID="JapaneseFood"/>

  <owl:Class rdf:ID="Lasagna">

    <rdfs:subClassOf>

      <owl:Class rdf:ID="ItalianFood"/>

    </rdfs:subClassOf>

  </owl:Class>

  <owl:Class rdf:ID="Salad">

    <rdfs:subClassOf rdf:resource="#ItalianFood"/>

  </owl:Class>

  <owl:Class rdf:ID="Soba">

    <rdfs:subClassOf rdf:resource="#JapaneseFood"/>

  </owl:Class>

  <owl:Class rdf:ID="Tenpura">

    <rdfs:subClassOf rdf:resource="#JapaneseFood"/>

  </owl:Class>
```

```
<owl:Class rdf:ID="KaKai">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ThaiFood"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Sushi">
  <rdfs:subClassOf rdf:resource="#JapaneseFood"/>
</owl:Class>
<owl:Class rdf:ID="GarlicBread">
  <rdfs:subClassOf rdf:resource="#ItalianFood"/>
</owl:Class>
<owl:Class rdf:ID="BaconTwist">
  <rdfs:subClassOf rdf:resource="#ItalianFood"/>
</owl:Class>
<owl:Class rdf:ID="Udon">
  <rdfs:subClassOf rdf:resource="#JapaneseFood"/>
</owl:Class>
<owl:Class rdf:ID="Okonomiyaki">
  <rdfs:subClassOf rdf:resource="#JapaneseFood"/>
</owl:Class>
<owl:Class rdf:ID="Papaya">
  <rdfs:subClassOf rdf:resource="#ThaiFood"/>
</owl:Class>
<owl:Class rdf:ID="TomYumKung">
  <rdfs:subClassOf rdf:resource="#ThaiFood"/>
</owl:Class>
<owl:Class rdf:ID="Pizza">
  <rdfs:subClassOf rdf:resource="#ItalianFood"/>
</owl:Class>
```

```
  <owl:Class rdf:ID="Yakisoba">
    <rdfs:subClassOf rdf:resource="#JapaneseFood"/>
  </owl:Class>
</rdf:RDF>


<!-- Created with Protege (with OWL Plugin 2.1, Build 284)
http://protege.stanford.edu -->
```

ข. ผลงานที่ตีพิมพ์จากงานวิจัย

A CONTEXT TYPE MODEL FOR CONTEXT-AWARE DISCOVERY OF WEB SERVICES

Aimrudee Jongtaveesataporn and Twittie Senivongse
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
Phyathai Road, Pathumwan, Bangkok, 10330 Thailand
Aimrudee.J@student.chula.ac.th, Twittie.S@chula.ac.th

ABSTRACT
A service registry such as UDDI is an important component for service discovery in an open Web services environment. Service discovery is refined if the registry is context-aware, i.e. realising the contexts of the querying service consumers as well as the contexts of the published services and service providers. Such awareness results in better matchmaking and service consumers will obtain query results that match better with their requirements. This paper presents an approach to enhance the standard UDDI registry with the context awareness capability. A context type model is proposed to model relevant contexts of the service consumers, service providers, and published Web services. The contexts can be static or dynamic and are associated with the semantics for context evaluation. Meanings of contexts are also considered, based on relevant ontologies, for better interpretation of the context information. With this approach, the context-aware UDDI can discover Web services whose contexts fit well to the contexts of the consumers. An architecture that supports the context type model is also presented.

KEY WORDS
Context, service discovery, UDDI, Web services, Web technology

1. Introduction

Web services technology has become the technology of choice for modern software applications with its ability to realise modular architectural design with services as software components that can be shared and composed in several applications. Web services architecture comprises three components – service providers, service consumers, and a service registry. Service providers can publish their details and the Web services they offer with a service registry called a UDDI registry [1]. As a result, service consumers can query from the UDDI for Web services that match with their requirements. UDDI information model is attribute-based; details about business entities (i.e. service providers) and business services (i.e. Web services) are described in terms of attribute names and attribute values. However, the information model accommodates only fundamental attributes such as business and service names, business address, and business and service categories. A service consumer may, for example, query for Web services under the category "online movie". Due to the simplicity of the UDDI information model, matchmaking can only give a rough set of candidate services, i.e. all online movie services which may provide different kinds of movie formats. This result set is rough because the service consumer's movie player software may be able to support only a particular movie format.

Contexts have increasingly been considered for better service provision. For instance, an online movie Web service which is aware of the consumer's device will adjust the movie quality according to the device. Dey describes a context-aware computing system as a system that uses contexts to provide relevant information and/or services to the users, where relevancy depends on the user tasks [2], while Korkea-aho defines contexts as any situational information that is available at the time of interaction between users and computing systems [3]. Examples of contexts are user identity or user profile, location, time, temperature, nearby people, operating device, software, heart rate, user activity, calendar information etc. Contexts play the role of a filtering mechanism, allowing only transmission of relevant data and services back to the service consumers so as to increase their satisfaction.

Contexts can be useful for service discovery also. The UDDI can better perform if contexts of service consumers, service providers, and Web services are considered at discovery time. As mentioned above, instead of finding all online movie Web services, the context-aware UDDI will consider and return only Web services that provide MPEG-4 movies if it is aware that the consumer has an MPEG-4 player software (i.e. Software context).

According to the definition in [2], relevancy of contexts varies from service to service. The Software context may be relevant to online movie Web services whereas they do not make sense for restaurant Web services. Since context awareness is service-specific and how the service utilises context information is up to the semantics of the service,

then how can context awareness be applied to a generic service such as the UDDI? To be context-aware, the UDDI requires a mechanism to recognise all contexts that are relevant to any published providers and services as well as any contexts of the consumers. It also has to know the evaluation semantics of these contexts in order to evaluate them properly during discovery. For example, in the case of online movie Web services, the Movie Category context has "equal" evaluation semantics. This means the context-aware UDDI will select the online movie services whose movie category is equal to the consumer's favourite movie category. On the other hand, the Load context has "minimum" evaluation semantics. This means the context-aware UDDI will return the online movie services with the minimum amount of load as a query result. To our knowledge, none of the researches on context-aware service discovery have elaborated on such a mechanism. In this paper, the mechanism is accomplished through a generic model of context information, namely a *context type model*.

Our context type model is effectively the model of context metadata including the definition and evaluation semantics of the contexts. Service consumers, service providers, and Web services have their context information described as *context attributes* under particular context types, and the context-aware UDDI will evaluate these context attributes during discovery time. A context attribute can be either static (i.e. unlikely to change its value) or dynamic (i.e. likely to change its value often). This is similar to the concept of service property type and static and dynamic service property in CORBA trader [4]. A standard UDDI will be made a context-aware UDDI with an extension to accommodate context types and context attributes to perform context-aware discovery. We also enhance the context type model by utilising OWL ontologies [5] as a knowledge base for richer interpretation of contextual information.

The rest of the paper is structured as follows. Section 2 reviews the related work. Sections 3 and 4 describe the context type model and the context attributes respectively. In Section 5, the architecture of our context-aware UDDI is presented. Section 6 discusses the approach and Section 7 concludes the paper with future research.

2. Related Work

Context awareness has been applied in Web services discovery researches. The WASP project [6] attempts to enhance the standard UDDI into UDDI+ by adding the semantic and contextual features. Their approach focuses on semantic analysis of service provider contexts which are described by the ontology-based DAML-S specification, and hence their contexts are static only. Doulkeridis *et al* [7] propose a context-aware service discovery architecture which accommodates various registry technologies including the UDDI and ebXML registry, but they consider the provision and consumption of services via mobile devices. Their approach therefore focuses on contexts related to mobility and handheld devices and does not cater for a generic context model. Lee *et al* [8] enhances context-aware discovery by introducing context attributes as part of service descriptions in the service registry but their contexts are dynamic attributes only. The CB-SeC framework [9] also enables more sophisticated discovery and composition of services, by having a WSDL of a Web service augmented with context functions; these context functions will be invoked to determine the values of the service contexts. In this way, however, the WSDL will be cluttered with operations that do not reflect service capability. Keidl *et al* [10] introduces the concept of context type in their context framework but they focus on adapting service provision according to the consumer's contexts which are specified under particular context types; their framework does not consider service discovery.

3. Context Type Model

Our approach views contexts as similar to service attributes which are represented by attribute names and corresponding values. Context values may be pre-specified by owners of the contexts (e.g. identity or profile of the consumer) or acquire on demand through a provided mechanism (e.g. a geographical location of the consumer via a GPS device). But unlike service attributes, the use of context information is implicit [11]. Services will determine the consumers' contexts as well as their own contexts implicitly and adapt their behaviour accordingly; the consumers do not have to specify explicitly what contexts the services should consider.

Since contexts can be realised by service attributes, we adopt the service property type and service property model of CORBA trader [4] for our context type model. With such a model, a service property type (i.e. service attribute type) gives the definition of the property (i.e. attribute) of a service. Similar services will share aspects characterised by the same service property types. In our approach, a context type gives the definition of a particular context as depicted in Figure 1.

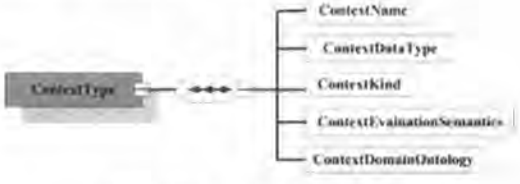

Figure 1. Scheme of context type.

A context type comprises the following information:

- **ContextName:** Each context type has a unique name. A context type may belong to a context category such as Zip Code is a location context.
- **ContextDataType:** A context type has an associated data type (i.e. an XML schema data type) for its value, e.g. string, integer, double, boolean etc.
- **ContextKind:** A context type can be either static (i.e. its value is unlikely to change) or dynamic (i.e. its value is likely to change often). The kind of context will affect how the context value is acquired.
- **ContextEvaluationSemantics:** A context type has associated semantics that guides how its value should be evaluated. This semantics is represented by usage operator, i.e. EQ (=), LT (<), GT (>), LE (<=), GE (>=), MIN, MAX.
- **ContextDomainOntology:** A context type may be associated with an external ontology that can help with the interpretation of the semantics of the context value.

Examples of context types are listed in Table 1.

Table 1 Examples of context types.

Context Name	Data Type	Kind	Evaluation Semantics	Domain Ontology
Location				
City	String	Static	EQ	Geographical Ontology
Zip Code	Integer	Static	EQ	-
Indoor/ Outdoor	Boolean	Static	EQ	-
Time				
Open Hour	String	Static	EQ	Temporal Ontology
Time Zone	String	Static	EQ	Temporal Ontology
QoS				
Load	Integer	Dynamic	MIN	-
Execution Duration	Double	Dynamic	MIN	-
Availability	Double	Dynamic	MAX	-
Bandwidth	Double	Dynamic	LE	-
Execution Cost	Double	Dynamic	MIN	-
Identity				
Credit Card	String	Static	EQ	-
Food Category	String	Static	EQ	Food Ontology
Software	String	static	EQ	Software Ontology

From the table, a context type City is defined to have a string value and is a static context. Its evaluation semantics EQ means the context-aware UDDI will implicitly check the service provider's city with the service consumer's city to see if they are equal. If they are not, the context-aware UDDI will interpret further the semantics of the context values by using a relevant external ontology (e.g. the service provider's city Beijing is the same as the consumer's city Peking according to a geographical ontology). The context type Load is associated with an integer value and is a dynamic context; its value will be dynamically determined at the time of context evaluation. Load is used with MIN evaluation semantics, meaning that the context-aware UDDI will discover the service with a minimum load. The context type Bandwidth is also a dynamic context but its evaluation semantics is LE; the context-aware UDDI will discover services whose bandwidth is less than or equal to the bandwidth of the consumer.

Context types will be defined by experts of the service domains or by service providers who make their services aware of such context types. A particular context type can be shared by several services and several consumers. For example, a context type City may be made aware by restaurant Web services and bookstore Web services. Several service consumers may also be equipped with a mechanism to provide their City context and hence can make a City-aware query for restaurant and bookstore Web services. Figure 2 shows the concept of context type sharing. Each Web service and service consumer is assumed to have a mechanism to acquire its context value from a certain context source.

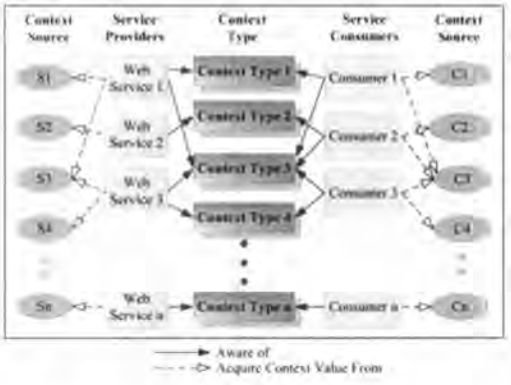

Figure 2. Sharing of context types.

4. Context Attribute

Context attributes are instances of context types. Service consumers have their context attributes associated with the query to the context-aware UDDI while service providers and their Web services have their context attributes published with the context-aware UDDI. A context attribute is modelled by

- **ContextName**: The context name identifies the context type to which the context attribute belongs.
- **ContextValue**: The context value is the data value according to the data type of the context type. It represents the context information. The context value is published with the context-aware UDDI if the context type is static. On the other hand, the context-aware UDDI will obtain the context value on demand from the service provider if the context type is dynamic.

- **ContextCallback**: If the kind of the context type is dynamic, the context-aware UDDI will need a way to acquire the context value of the service dynamically. This is by invoking a function of the service provider's callback interface at the specified URL.

5. System Architecture

The architecture for context-aware service discovery concerns the provision for consumer-side context information, the provision for provider-side context information, and the context-aware UDDI.

In this paper, we do not focus on the acquisition of context information from context sources. It is assumed that service consumers and service providers have mechanisms to obtain their context information from any context sources and provide to the context-aware UDDI. A service consumer may have a consumer-side context manager whose job is to acquire the consumer's contexts, either static or dynamic, from any context sources and attach such information as context attributes with the query request to the context-aware UDDI. The consumer's context attributes may be transmitted in a SOAP header of the query, such as in [10].

The emphasis of this paper is on the context-aware UDDI or the Ctx-UDDI. The basis of its architecture is the extension to UDDI so that it can accommodate publishing of additional service attributes in business entities and business services, but these service attributes will serve as context attributes that will be considered implicitly at discovery time. The architecture is shown in Figure 3 and comprises the following:

- **Extended UDDI API** is the API for querying and publishing with the Ctx-UDDI. It also bridges the context-aware feature with the standard UDDI registry. It allows for publishing of context types and service providers' context attributes. Standard service descriptions are sent to publish via the standard UDDI API, and additional context attributes are sent to store with the provider-side context repository. The extended UDDI API accepts standard UDDI query and sends it to the standard UDDI API, as well as accepts context-aware query. The extended UDDI API is implemented using Java technology.
- **Standard UDDI API** handles the standard service publishing and querying. jUDDI API is used for the implementation.
- **Standard UDDI database** stores the standard service descriptions.
- **Provider-side context manager** manages published context types and allows service providers to register which context types they and their Web services are aware. Similarly to the provision for consumer-side context information, the provider-side context manager assumes the service providers have a mechanism to acquire their context information. In the case that the contexts are static and are published in the provider-side context repository, the provider-side context manager can query directly for those values. On the other hand, the provider-side context manager will invoke callback interfaces of the service providers to obtain dynamic context values. The callback interface is an interface with a generic function for retrieving the value of a dynamic context. Service providers will implement this interface and publish the URLs with the provider-side context manager. Static and dynamic context values will be sent to evaluate at the service matching module.
- **Provider-side context repository** stores published context types and service providers' context attributes.

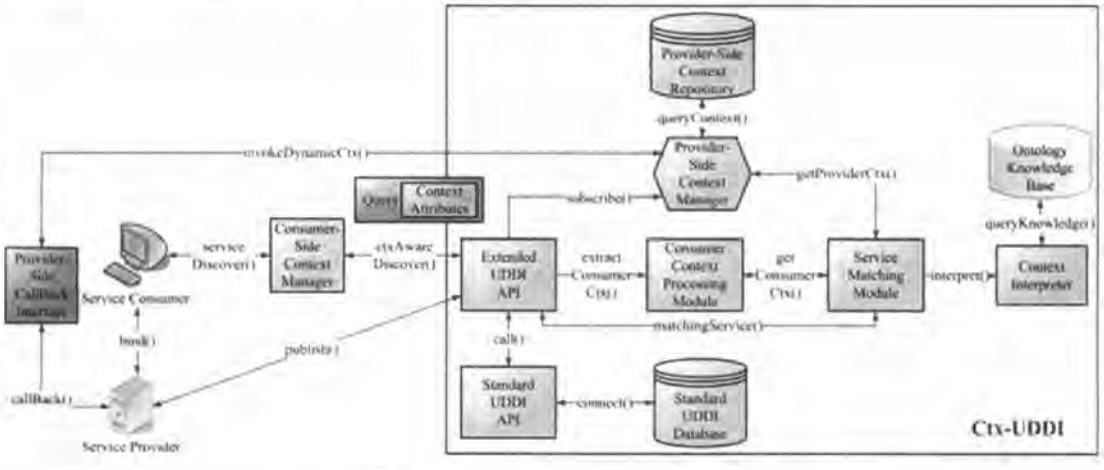

Figure 3. Context-aware service discovery architecture.

- **Consumer context processing module** extracts service consumers' context attributes from SOAP queries.
- **Service matching module** evaluates context information of the candidate services that are the preliminary results of standard UDDI query. It selects Web services that will best fit the consumers' contexts.
- **Context interpreter** interprets the semantics of both service providers' contexts and service consumers' contexts by using the semantic knowledge in the ontology knowledge base.
- **Ontology knowledge base** is the database of semantic knowledge that is derived from external domain ontologies that are relevant to the published context types.

5.1 Publishing with Ctx-UDDI

In service provider's view, publishing messages that are sent to the Ctx-UDDI can be:

- **Standard publishing:** Service providers do not specify which context types they and their Web services are aware. The Ctx-UDDI accepts the publishing messages and stores business and service descriptions in the standard UDDI database.
- **Context-aware publishing:** Service providers specify what context types they and their Web services are aware, and publish context attributes. Standard business and service descriptions are stored in the standard UDDI database while context attributes are stored in the provider-side context repository. Information entries in these two databases are linked together via the same businessKey and serviceKey.

5.2 Querying with Ctx-UDDI

In service consumer's view, querying messages that are sent to the Ctx-UDDI can be:

- **Standard query:** If no context attributes are attached to the service consumer's request, the Ctx-UDDI accepts this standard query and discover from the standard UDDI database.
- **Context-aware query:** If context attributes are attached to the service consumer's query, the consumer's context will be extracted and then the query will be made on the standard UDDI database. The resulting candidate services will have their contexts evaluated further. The provider-side context manager finds the contexts attributes of these services in the provider-side context repository by using the linked businessKey and serviceKey. The context values will be determined if they fit to the given consumer's context information. Matched services will be returned to the consumer.

5.3 Matchmaking by Ctx-UDDI

Two kinds of context-aware matchmaking are supported:

- **Match by evaluation semantics:** The provider-side contexts will be evaluated against the consumer's contexts according to the evaluation semantics of the relevant context types. The following scenarios refer to the context types in Table 1.

Scenario#1. Suppose a restaurant Web service is aware of the context type City and has published the context value Bangkok. Since the evaluation semantics of City is EQ, the Ctx-UDDI will return only the restaurant Web services that are situated in the same city as the service consumer. That is, if the service consumer has a City context with the value Bangkok attached to its query, this restaurant Web service will be a match result. In this scenario, it might be convenient for the service consumer to interact (e.g. make a reservation) with the restaurant services within the same city.

Scenario#2. Suppose a service consumer is looking for an online movie Web service. The query has a context attribute Bandwidth with the value 56 kbps attached (this is the capacity of the consumer's modem). Since the evaluation semantics of Bandwidth is LE, the Ctx-UDDI will return only the online movie Web services with the context Bandwith less than or equal to 56 kbps. In this scenario, the consumer is prevented from experiencing possible long delay and jitter if connected to the services with a bandwidth higher than the capability of the consumer's device.

- **Match by ontology**: The semantics of the provider-side contexts and the consumer's contexts will be evaluated according to an external ontology that is relevant to the context type. The following scenario refers to the context type in Table 1.

Scenario#3. Suppose a service consumer is looking for an online movie Web service. The query has a context attribute Software with the value QuickTime attached; it is the movie player software installed in the consumer's device. An online movie Web service provides MPEG4 movie files and is published to be aware of MPEG4Player software. By evaluation semantics EQ, the service's MPEG4Player context is not equal to the consumer's QuickTime context. However, by an external ontology about software (Figure 4), QuickTime is a kind of MPEG4Player. Therefore the Ctx-UDDI will interpret that this online movie Web service is a match to the query.

Figure 4. Part of software ontology.

6. Discussion

Our approach assumes service providers and service consumers follow the context type model. Each context type defines semantics of a particular context, and hence several providers' Web services and several consumers can make themselves aware of it. It is possible that service providers may browse the list of published context types and select to provide their services with certain context-aware features. Similarly, service consumers may browse the list of published context types and prepare the mechanisms to attach certain contexts to their requests.

The Ctx-UDDI does not assume that the service consumers are to be aware of all contexts supported by the Web services in order to make context-aware queries. It is difficult to assume that all consumers will have the capability to support context awareness (e.g. have a GPS device or other mechanism for the location context) in order to query for any Web services that may exhibit context-aware features (e.g. restaurant or any other Web services that are location-aware). Table 2 shows how the Ctx-UDDI considers Web services against the query. Suppose a query is attached with context attributes of types A and B, and several Web services are aware of various context types as in the table. It can be viewed that the consumer side would like to enable context-aware features that relate to context types A and B. Thus the Ctx-UDDI will try to find Web services that support at least one of the context types attached with the query; this is best-effort discovery. In this case, Web services 1, 2, and 3 are considered against the query. On the contrary, Web services 4 and 5 do not exhibit the requested context-aware features and will not be considered.

Table 2. Matching consideration.

	Context Type		
	A	B	C
Query	✓	✓	
Web service 1	✓	✓	✓
Web service 2	✓		✓
Web service 3		✓	
Web service 4			✓
Web service 5			
✓ aware of			

7. Conclusion

The contribution of this paper is the context type model that gives definitions to context types in the evaluation of consumer-side and provider-side contexts in service discovery. The discovery architecture realises context types and context attributes by static and dynamic service attributes which are the extension to the standard UDDI information model. Matchmaking process implicitly considers consumer-side against provider-side contexts according to the evaluation semantics and ontological semantics of the contexts.

As for future work, we plan to work on ranking of matched results and study the performance of discovery. The context type model can also be improved. Relationships between context types may be defined, e.g. a context type may be a subtype of another. This will give flexibility to the evaluation of the context attributes.

Acknowledgement

This research is part of the Engineering New Paradigm Software for Enterprises with Service-Oriented Architecture Project, supported by Thailand's Software Industry Promotion Agency (Public Organisation).

References

[1] uddi.org, UDDI, Available from: http://www.uddi.org

[2] A. Dey, *Providing architectural support for building context-aware application, Ph.D. dissertation* (Atlanta: Georgia Institute of Technology, 2000).

[3] M. Korkea-aho, Context-aware applications survey, *Internetworking Seminar (Tik-110.551)*, Helsinki University of Technology, Spring 2000. Available from: http://users.tkk.fi/~mkorkeaa/doc/context-aware.html

[4] Object Management Group, *Trading Object Service Specification*, Revised Edition, March 1997.

[5] w3.org, OWL Web Ontology Language Overview, Available from: http://www.w3.org/TR/2004/REC-owl-features-20040210/

[6] S. Pokraev, J. Koolwaaij and M. Wibbels, Extending UDDI with context-aware features based on semantic service descriptions, *Proc. of 1st Intl. Conf. on Web Services*, Las Vegas, Nevada, USA, 2003, 184-190.

[7] C. Doulkeridis, N. Loutas, & M. Vazirgiannis, A system architecture for context-aware service discovery, *Proc. of Intl. Workshop on Context for Web Services (CWS'05)*, Paris, France, July 5, 2005, 101-116.

[8] C. Lee & S. Helal, Context attributes: An approach to enable context-awareness for services discovery, *Proc. of Symposium on Application and the Internet*, Florida, USA, 2003, 22-30.

[9] S. K. Mostefaoui, H. Gassert, & B. Hirsbrunner, Context meets web services: Enhancing WSDL with context-aware features, *Proc. of 1st Intl. Workshop on Best Practices and Methodologies in Service-Oriented Architectures: Paving the Way to Web-services Success*, Vancouver, British Columbia, Canada, 2004, 1-14.

[10] M. Keidl & A. Kemper, Towards context-aware adaptable web services, *Proc. of 13th Intl. World Wide Web Conf. - Alternate Track Papers & Posters*, New York, USA, 2004, 55-65.

[11] C. Doulkeridis & M. Vazirgiannis, Querying and updating a context-aware service directory in mobile environments, *Proc. of 2004 IEEE/WIC/ACM Web Intelligence Conf.*, Beijing, China, 2004, 562-565.

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวเอมฤดี จงทวีสถาพร เกิดเมื่อวันที่ 21 มกราคม พ.ศ. 2526 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับประถมศึกษาและมัธยมศึกษาตอนต้นจากโรงเรียนอัสสัมชัญคอนแวนต์ จากนั้นเข้าศึกษาต่อระดับมัธยมศึกษาตอนปลายที่โรงเรียนเตรียมอุดมศึกษา จนจบการศึกษาระดับมัธยมศึกษาตอนปลายเมื่อปีการศึกษา 2544 และสำเร็จการศึกษาระดับปริญญาบัณฑิต ในสาขาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2548