Towards a Language Service Infrastructure for Mobile Environments

Nguyen Cao Hong Ngoc, Donghui Lin, Takao Nakaguchi, and Toru Ishida
Department of Social Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-Ku, Kyoto, 606-8501, Japan
nchngoc@ai.soc.i.kyoto-u.ac.jp, {lindh, nakaguchi, ishida}@i.kyoto-u.ac.jp

Abstract

Since mobile devices have feature-rich configurations and provide diverse functions, the use of mobile devices combined with the language resources of cloud environments is high promising for achieving a wide range communication that goes beyond the current language barrier. However, there are mismatches between using resources of mobile devices and services in the cloud such as the different communication protocol and different input and output methods. In this paper, we propose a language service infrastructure for mobile environments to combine these services. The proposed language service infrastructure allows users to use and mashup existing language resources on both cloud environments and their mobile devices. Furthermore, it allows users to flexibly use services in the cloud or services on mobile devices in their composite service without implementing several different composite services that have the same functionality. A case study of Mobile Shopping Translation System using both a service in the cloud (translation service) and services on mobile devices (Bluetooth low energy (BLE) service and text-to-speech service) is introduced.

Keywords: language service infrastructure, mobile environment, service composition

1. Introduction

Due to the developments of both transportation systems and the Internet, geographical distance is no longer a barrier for people to communicate with each other (Cairncross, 2001). Intercultural and multilingual activities are occurring frequently in daily life (Kim, 2000). However, differences in language and culture remain critical issues (Sechrest et al., 1972) (McGorry, 2000) (Scollon et al., 2011) (Piller, 2011), that raise difficulties in understanding each other, and limit us from achieving the fullest communication expected. Recent significant advances in technology have made mobile devices such as smart phones and tablets to popular in daily life. The advantages of mobile devices should make it possible to overcome language barriers in communication.

Consider the scenario of a Japanese girl traveling to Vietnam with an American. While walking on the street, her smart phone receives notification of a special discount at a Vietnamese restaurant. The notification and further information on her phone are in Japanese while her friend, the American, receives the same notification but in English. It is very convenient for them because they do not know Vietnamese and the local are not good at English or Japanese. It is good for both travelers and the businesses since without the notification and translation, travelers may face the linguistic barrier and the restaurant may miss out on potential customers.

To solve the similar problems, the concept of mobile cloud computing (Qureshi et al., 2011) (Bahl et al., 2012) (Khan et al., 2014), where the mobile device can use the cloud for data processing, has been proposed. Several mobile cloud applications are available, for example Google Map, Gmail for iPhone, Cisco's WebEx on iPad. To implement such applications, developers must have good knowledge of both cloud computing techniques and mobile device programming techniques. In addition, critical issues of composing, and integrating different types of services need to be solved. However, these tasks are not easy for all developers. One of the promising solutions is using a language service infrastructure, which provides language resources and integrates services in mobile devices. By using the language service infrastructure for mobile environments, users can use both existing language resources in the cloud and resources of mobile devices, and create their own services. There are existing language service infrastructures such as Language Grid (Ishida, 2006), CLARIN (Váradi et al., 2008), PANACEA (Bel, 2010), Meta-Share (Piperidis, 2012), and LAPPS (Ide et al., 2014), however none of them support for both services in the cloud (hereafter called cloud services) and services on mobile devices (hereafter called mobile services). In this research, we achieve our goal by extending the Language Grid, which is the language service infrastructure for cloud environments, to cover mobile environments.

Since services in the cloud and services on mobile devices have different communication protocols and different input and output methods, establishing the language service infrastructure for mobile environments faces two main issues: (1) for the atomic services: utilizing both services in the cloud and services on mobile devices (including data on mobile devices and device functions wrapped as services), and (2) for the composite services: handling the differences of services in the cloud and services on mobile devices and managing the different execution of services in the two environments.

The remainder of this paper is organized as follows: Section 2 explains the reason why the language service infrastructure for mobile environments is needed and introduces our design concept. The architecture of our language service infrastructure is presented in Section 3. Section 4 describes the realization of the proposed language service infrastructure. A case study on the use of the proposed language service infrastructure is shown in Section 5. Section 6 describes the related work. Finally, Section 7 concludes this paper.

2. Language Service Infrastructure for Mobile Environments

2.1. Why Language Service Infrastructure for Mobile Environments?

Since 2006, Language Grid (Ishida, 2006) (Murakami et al., 2010) (Ishida, 2011) (Ishida et al., 2012), the language service infrastructure, has been operated. The language resources are wrapped as services and provided those services to users. Although Language Grid is the language infrastructure supporting people in intercultural and multilingual activities, Language Grid provides only cloud language services and the infrastructure to combine them. There are many scenarios in which the combination of both services in the cloud and services on mobile devices will bring many benefit to users.

With the rapid growth in hardware complexity and services, mobile devices now offer many sophisticated functions such as speech recognition, text-to-speech, Bluetooth low energy (BLE) links, gesture recognition, camera, text service. The use of mobile devices to assist people in overcoming language barriers is becoming more and more popular such as translation via an image of unknown word, the spoken word, and gestures. There are many such applications that take advantage of the display screen, microphone, speaker, camera, gesture which are unique functions in mobile devices for translation. The use of available functions in mobile devices as well as the cloud services brings us several advantages such as providing more alternatives in selecting a service, and utilizing unique functions that are used on-site as sensor services (for example the camera, gesture, BLE). These functions cannot be replaced by cloud services.

Thus, the use of resources and services on mobile devices as well as resources and services in the cloud is essential. However, currently, there is no infrastructure that can support those tasks easily. Each developer has to find his or her own solution for creating such applications, which is time consuming. A language service infrastructure for mobile environments is required.

2.2. Design Concept

Given the necessity of combining cloud services and mobile services, we need a language service infrastructure that provides language resources and can integrate with mobile services so that users can create and use their own services. Using this infrastructure, users can invoke existing language resources which are registered on the language platform such as Google translation service, Bing translation services, Baidu translation service, SYSTRANet, Standford POS Tagger, SVMTool. Moreover, users can compose those existing resources with not only their own language resources in their mobile devices but also their mobile device functions which already wrapped as services to create their required composite services. The language service infrastructure for mobile environments that fulfills our design concept is described in Figure 1:

- Resource: includes the mobile device resources such as device functions and user's data (for example: parallel texts) and the existing cloud resources.

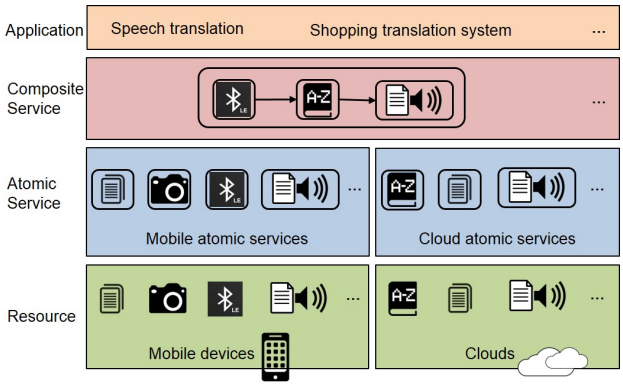

Figure 1: Design concept

- Atomic service: includes Mobile Atomic Services and registered language services in the cloud. Mobile Atomic Services can be atomic languages services on mobile devices or mobile device functions wrapped as services.
- Composite service: Atomic services can be composed by the Web service workflows. A services described by a workflow is called a composite service. Various composite services can be made, including the mobile services and cloud services.
- Application: can be any application which utilize the composite service via the language service infrastructure on mobile devices.

3. Architecture

The main idea of this design concept is an appropriate infrastructure to combine the services in the cloud and services on mobile devices. However, these services have different characteristics, for instance, the input and output method, the communication protocol. While services in the cloud usually require input and output parameters, services on mobile devices sometimes does not require all those parameters because service on mobile devices use device components as input and output. The input component can be the microphone, the camera, the geoloc, the compag, the gesture; the output component can be the display screen, the speaker, the vibration sensor. With regard to the communication protocol, mobile services use intra-device connections while cloud services need an Internet protocol. Therefore, our proposed framework includes appropriate modules to deal with those issues.

3.1. Overall Framework

The language service infrastructure for mobile environments consists of four main components: Service Supervisor, Grid Composer, Service Database and Service Container. The idea of Grid Composer Component and Service Database Component are those of the Language Grid (Murakami et al., 2011) but some unnecessary modules are omitted. In fact, Service Supervisor Component Service Container Component are modified.

- Service Supervisor: controls service invocation by service users. This component is in charge of both inter-grid and intra-grid execution of services.

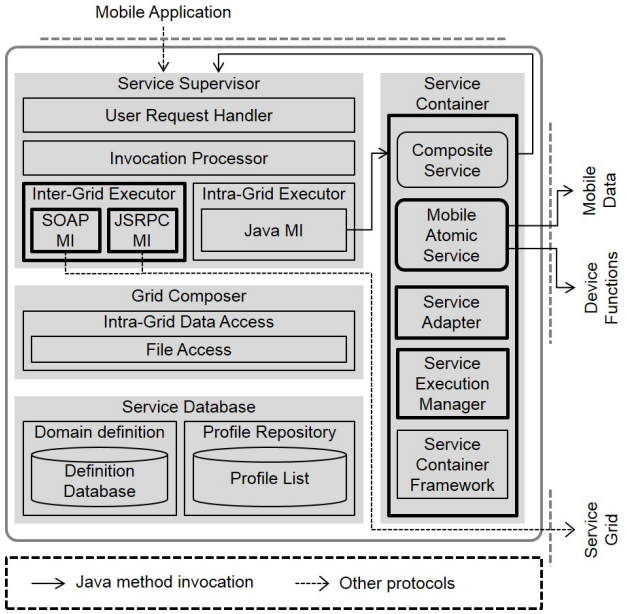

Figure 2: Overall framework

- Grid Composer: creates a grid network within its services grid. Though Language Grid supports inter-grid network to share service information across grids, our framework does not support it and we omit the inter-grid data access module.
- Service Database: is a repository to store various types of service information.
- Service Container: consists of all services such as composite services, mobile atomic services, and some modules which control the execution of those services.

Figure 2 describes the overall framework of the language service infrastructure for mobile environments.

3.2. Service Supervisor

The *Service Supervisor* controls service invocation by service users. Similar to Language Grid, the control covers access endpoint locating, and access logging. The *User Request Handler* extracts from the service request the information necessary to invoke a service. The extracted information is sent to the *Invocation Processor*, which executes the sequence of pre-process, service invocation, post-process, and logging.

In order to establish method invokers for both cloud services and mobile services, both *Inter-Grid Executor* and *Intra-Grid Executor* are needed. The *Intra-Grid Executor* is responsible for creating method invokers for services on *Service Container* while the module *Inter-Grid Executor* is responsible for creating method invokers for cloud services. In *Inter-Grid Executor*, *SOAP Method Invoker* (SOAP MI) uses SOAP to communicate with services while *JSRPC MI* uses JSON-RPC. *SOAP MI* encodes a method invocation request as a SOAP request and decodes SOAP response to return object. *SOAP MI* and *JSRPC MI* have same functionality as in the original Service Grid.

This component is modified to deal with the differences in communication method between cloud services and services on the mobile services. For example, in order to use the speech recognition service in mobile devices, *Java Method Invoker* (Java MI) is used while in order to use the translation service in the cloud such as Google translation service, *SOAP MI* is used. The composite service that consists of speech recognition service and Google translation service for translating the voice message to another language needs to satisfy the different communication methods of its two atomic services. Therefore, *Intra-Grid Executor* as well as *Inter-Grid Executor* are played the important role in our framework.

3.3. Grid Composer

The *Intra-Grid Data Access* provides read and write interfaces for the *Service Database*. Though we actually do not need to compose grid because we focus on combining mobile services and cloud services inside mobile devices, we keep the *Grid Composer* module so that we will be able to support inter-grid connection in the future.

3.4. Service Container

The *Service Container* executes composite services and atomic services. The *Mobile Atomic Service* that executes atomic services by wrapping resources (including mobile data and device functions) as services with standard interfaces. The *Composite Service* provides service workflow execution, and dynamic service binding defined as system requirements. The *Service Adapter* is designed to deal with the adaptation of device functions wrapped as services on mobile devices to suit ordinal cloud services. The *Service Execution Manager* has responsibility of executing *Composite Services* (cloud services combined with mobile atomic services), especially some monitoring or event-detecting services. These services run continuously to detect events and have their own life cycles whereas normal services work once when invoked. Therefore, the composite services that contain the monitoring function must be run constantly. For example, the BLE detection service, which uses Bluetooth connection to detect the nearby beacons and returns the beacon list, needs to update beacon information every second. *Service Execution Manager* supports this kind of execution. To deal with event-detecting services without the event processing middle-ware, we simplify the execution model by introducing periodical execution of composite services and a queue for each event-detecting service instead of centralizing on shared event-queue middle-ware.

4. Realization of the Language Service Infrastructure for Mobile Environments

Implementing the language service infrastructure for mobile environments requires two main issues to be solved: (1) for the atomic services: utilizing both services in the cloud and services on mobile devices (including mobile data and device functions wrapped as services), and (2) for the composite services: handling the differences of service in the cloud and services on mobile devices and the different execution demands of services in the cloud and services on mobile devices. Our proposed framework provides some special modules including *Inter-Grid Executor*, *Mobile Atomic Services*, *Service Adapter*, *Service Execution Manager* in

Service Interface	Input	Output	Function
DeviceSpeechRecognitionService	Language	Text	Receive speech via the microphone and convert to text
DeviceTextToSpeechService	Language, Text, VoiceType, AudioType	N/A	Read a text
DeviceGestureService	Language	Text	Recognize text from hand-writing on screen
DeviceLocationService	N/A	Position	Return current position (latitude, longitude)
DeviceTextService	Language	ListOfSuggestion	Check spelling
DeviceBLEService	N/A	ListOfBeacons	Detect nearby beacons
DeviceCameraService	CaptureRequest	CaptureResult	Capture a single image from the image sensor

Table 1: Interfaces of device functions

order to fulfill these requirements. However, in the initial phase, we only force on the most important parts to realize the language service infrastructure. They are the *Mobile Atomic Services* and the *Service Adaptation*.

4.1. Mobile Atomic Services

This is one module of *Service Container* component and consists of mobile data wrapped as services and device functions wrapped as services. Mobile data can include user's dictionaries or parallel texts or any user data which can be wrapped as services with wrappers compatible wrappers with our framework. In order to utilize the device functions, the compatible interfaces (Table 3.4.) are provided using device native APIs.

4.2. Service Adapter

This is one module of *Service Container* component to deal with the adaptation of device functions wrapped as services on mobile devices. Since services in the cloud usually require input and output parameters, services on mobile devices sometimes does not require all those parameters (because service on mobile devices use device components as input and output), some services which have the same functionality in the cloud and on mobile devices have different interfaces. For instance, the interface of speech recognition service in Language Grid has *speech* as input while the interface of speech recognition service in mobile devices does not. This complicates the developers' task when they create the composite service. For each concrete service, they have to write one composite service that corresponds to that service's interface. It is more convenient for developers if they need to write just one composite service and have it bound to many concrete services that have the same functionality. Service adapters in the framework are the bridge between the developer view and concrete services. Service adapters can handle:

- the difference in service names (for example: the composite service requires *DeviceTextToSpeechService* but the concrete service is *TextToSpeechService*)
- the difference in service input and output parameters (for example: *DeviceTextToSpeechService* requires only language as input parameter but *TextToSpeechService* requires both *language* and *speech* as input parameters)

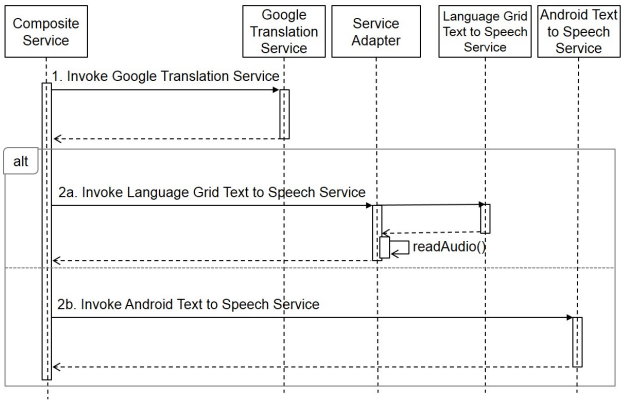

Figure 3: An example of interface adaptation process of text-to-speech translation service

Our proposed solution is to select the *Service Adapter* automatically by *Service Container* when the adaptation is needed. When a composite service combines *TranslationService* and *DeviceTextToSpeechService* and the concrete service of *DeviceTextToSpeechService* is the cloud service, the *Service Container* assign the *Service Adapter* to the target of invocation of *DeviceTextToSpeechService* from the composite service (Figure 3). In this example, *DeviceTextToSpeechService* is required but the concrete service is *TextToSpeechService* so the service names are obviously different. Moreover, the text-to-speech service on Android devices does not need the output parameter (because the speech can go directly through the speaker) whereas the text-to-speech service in Language Grid requires the speech as the output parameter. Therefore, *Service Adapter* has to deal with two differences (the difference in service names and the difference in service input and output parameters). The different in service name is fixed in the process of returning the actual wrapper by *Service Container*. Instead of finding service by types, *Service Container* will find all services and return the compatible service by using the mapping table. The different in service input and output parameters is fixed by adding more functions corresponding to each service adapter. For example, the adapter for *DeviceTextToSpeechService* needs a function to read the audio file from the output of *TextToSpeechService* on the mobile device. Table 4.2. shows some examples of how to solve different interfaces between mobile services and cloud services.

Adapter Name	Service Interface (Android)	Service Interface (Language Grid)	Description
DSRStoSRSAdapter	public interface **DeviceSpeechRecognitionService** { String recognize (String language); }	public interface **SpeechRecognitionService** { String recognize (String language, **Speech speech**); String[] **getSupportedVoiceTypes**(); String[] **getSupportedAudioTypes**(); }	This adapter implements the adaptation of speech recognition service in Language Grid to the speech recognition service on Android devices. A function to record an audio file from the microphone to be the input of speech recognition service in Language Grid is required.
DTTSStoTTSSAdapter	public interface **DeviceTextToSpeechService** { **void** speak (String language, String text, String voiceType, String audioType); . . . }	public interface **TextToSpeechService** { **Speech** speak (String language, String text, String voiceType, String audioType); . . . }	This adapter implements the adaptation of text-to-speech service in Language Grid to the text-to-speech service on Android devices. A function to play an audio file, which is the output of text-to-speech service in Language Grid, is required.

Table 2: Examples of interface adaptation of mobile services (Android) and cloud services (Language Grid)

5. Case Study

In this section, we will introduce some examples which use both services in the cloud and services on mobile devices to support people overcome the language barrier in intercultural and multilingual environments.

5.1. Mobile Shopping Translation System

As mentioned in the scenario in the Introduction, the store in the system will have a beacon that advertises the store's information. When travelers across the store, their mobile devices (running the application of this system) will receive notification from the beacon. Travelers can view on the mobile or listen via their headphone the summary or detailed information from the stores in their own language before making decision to go inside the store or not. The beacon only advertises the store's information; other information will be transferred between the system server and the user's mobile device associated with existing translation services provided in the cloud. The stores can promote their information in any language, regardless of customer's languages and the language translation is done transparently to the customer. The customer can choose the language desired.
In Mobile Shopping Translation System, composite service is a combination of BLE service, translation service and text-to-speech service. BLE service is a service on mobile devices. Translation service can be any provided translation service in the cloud such as Bing translation, Google translation, Baidu translation. Text-to-speech service can be a service in the cloud (for example text-to-speech service of Language Grid) or a service on mobile device (for example text-to-speech function on Android device wrapped as a service). This example application shows the necessary of composing cloud services and mobile services. Without the proposed language service infrastructure, it is more complicated to combine the BLE function, the text-to-speech function with the translation service in the cloud since the device functions have not realized as services. In addition, developers have to write two different composite services, one for the text-to-speech function on the device and one

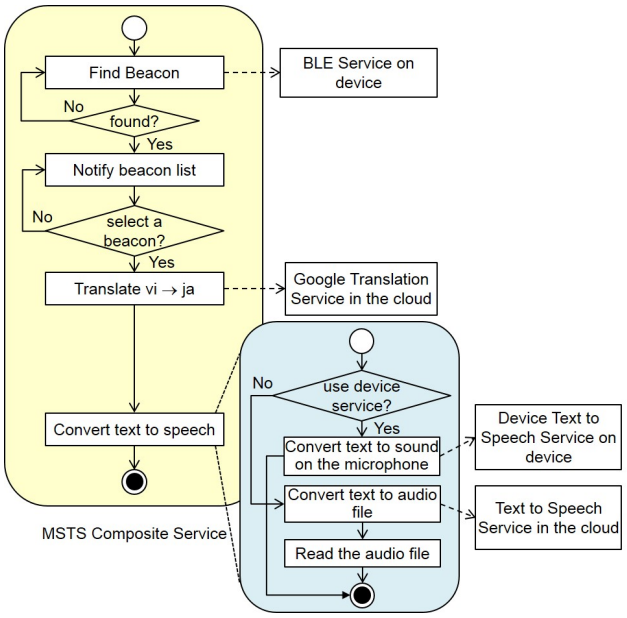

Figure 4: The composite service of Mobile Shopping Translation System (MSTS)

for the text-to-speech service in the cloud. Moreover, if someone else has to develop other application which requires BLE function or text-to-speech function, he or she has to write everything from the beginning. Thanks to the proposed language service infrastructure, BLE function and text-to-speech function are realized as services, developers easily combine them with other services and they need to write only one composite service which can use both text-to-speech service in the cloud and text-to-speech service on mobile devices (Figure 4).

The language service infrastructure calls the *ServiceContainer.java* file of the *Service Container*, method *findAll()*, which returns all types of services, while the *BindingServiceFactory.java* (Figure 5) is called to check if the composite service needs *DeviceTextToSpeechService* while the wrapper provides *TextToSpeechService*; the adapter must

ServiceContainer.java

```
package org.langrid.servicecontainer;
...
public Collection<String> listBindableServices(final String serviceType,
final String execType) {
  ...
  List<MobileService> wrappers =
    getDaoFactory().createMobileServiceDao().findAll();
  ...
}
...
```

BindingServiceFactory.java

```
package org.langrid.servicecontainer.compositeservice;
...
public <T> T getService(String invocationName, Class<T> interfaceClass) {
  ...
  Class<?> clazz = wsw.loadClass(wrapper.getName());
  T s = (T) clazz.newInstance();
  for (Map.Entry<String, String> setting:wrapper.getSettings().entrySet()) {
    ...
  }
  if (interfaceClass.isAssignableFrom(clazz)){
    return (T)s;
  } else {
    return findAdapter(interfaceClass, clazz, s);
  }
  ...
}
```

Figure 5: Classes related with interface adaptation process in Mobile Shopping Translation System application

download the audio file to the device and play it because the result of the actual wrapper has no output parameter.
In *BindingServiceFactory.java* file of the *Composite Service*, the variable *interfaceClass* is the interface that the composite service will invoke. The variable *clazz* is the class of wrapper. The variable *s* is wrapper instance. What adapter is required to do is convert the interface that *clazz* implements to *interfaceClass.findAdapter()* function is implemented as a table-lookup process.

5.2. Other Applications

There are more interesting applications that can be created as combinations of services in the cloud and services on mobile devices. People have already developed some initial applications; but they have to develop everything by themselves. Due to our proposed infrastructure, their tasks become easier because they only have to combine the existing services in the cloud and on mobile devices. Our proposed infrastructure has covered most of the main issues in more fully utilizing the power of cloud services and mobile services.

- Speech translation application: This application receives speech via device's microphone, converts to text, translates it into target language and output the translated text as speech via the device's loudspeaker. The speech recognition service, translation service and text-to-speech service are required. However, both mobile device and the cloud provide the speech recognition service and text-to-speech service. To utilize both cloud and mobile services, two adapting functions are needed: one function to record an audio file from the microphone and pass it as input to the speech recognition service in the cloud and one function to play an audio file, which is the output of text-to-speech service in the cloud.

- User manual from the bar code application: The main idea of this application is to use device's camera to scan a bar code then retrieve and show the corresponding user manual (and price). Therefore, the composite service includes the camera service, user manual corpus. The camera service is a mobile service while user manual corpus is the service in the cloud. The translation service in the cloud and user dictionaries on the mobile device may be integrated if necessary.

- Photo wiki application: The main idea of this application is to use device's camera to scan a word of interest to the user and return the wiki of that word. The camera service, image-to-text service and wiki resource are three main services in the composite service needed for this application.

6. Related Work

Numerous of platforms have been proposed for creating, coordinating and making language resources available and readily usable for users. They are Language Grid (Ishida, 2006), PANACEA (Bel, 2010), Meta-Share (Piperidis, 2012), and LAPPS (Ide et al., 2014). None of these, unfortunately, satisfy the requirements of utilizing services in the cloud and device functions on mobile devices, none of them is satisfied. Language Grid, PANACEA, Meta-Share, LAPPS provide language resources and language tools and some of them provide a workflow manager or composition tools but all of them are web-service based. None of those platforms offer services targeting mobile device function. The proposed framework can utilize services on both cloud and mobile devices, as well as supporting the combination among them.
Language Grid (Ishida, 2006), Meta-Share (Piperidis, 2012) define their own standards to ensure the compatibility of services. The core of the Meta-Share model is the *resourceInfo* component, which contains all the information relevant for the description of a language resource. LAPPS (Ide et al., 2014) defines a *Web Service Exchange Vocabulary* and established a *Web Service Exchange Vocabulary Repository*. Each service in the LAPPS Grid publishes metadata describing what it requires for input and what it produces as output. *Web Service Exchange Vocabulary* specifies a terminology for a core of linguistic objects and features exchanged among natural language processing tools. That is, it not only identifies a standard terminology but also indicates the relations among them. PANACEA (Bel, 2010) is grounded on the use of existing standards for defining web service input and output formats, i.e. Traveling Objects. It has implemented converters to reduce the bulk of input and output format problem. Our proposed framework aims to solve the differences between the interfaces of services on cloud and services in mobile devices. Those services have the same functionality but require different input and output parameters since mobile devices can interact directly with users via their device components such as the microphone or speaker. Thanks to the *Service Adapter* module in our proposed framework, this issue is

resolved so that developers need not to write different composite services for the same functional service on different devices (mobile device or located servers).

7. Conclusion

This paper proposed a language service infrastructure for mobile environments. It opens a new approach to overcome the language barrier with mobile devices. Our proposed infrastructure use both *Inter-Grid Executor* Module and *Intra-Grid Executor* Module to overcome the different of communication protocols between cloud services and mobile services; *Service Adapter* deals with the different in input and output methods between cloud services and mobile services; and *Service Execution Manager* deals with the execution of the composite service, especially those that includes the monitoring services. The proposed infrastructure not only allows users to use existing language resources in the cloud as well as their own language resources on their mobile devices, but also allows users mashup cloud services and mobile services (services on the devices and device functions wrapped as services). In addition, it allows users to flexibly combine cloud services and/ or mobile services in their composite services without implementing many different composite services which have the same functionality. Finally, a case study of Mobile Shopping Translation System was presented to illustrate our proposed language service infrastructure.

Since some device functions place restriction on threads (for example, some of them need to be invoked in user interface (UI) thread or non-UI thread) or exclusive execution (for example, some of them can be executed only from a single client), the framework should deal with this issue. This task is considered to be our future work.

8. Acknowledgements

This research was supported by a Grant-in-Aid for Scientific Research (S) (24220002, 2012-2016) from Japan Society for the Promotion of Science (JSPS).

9. Bibliographical References

Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan. 2012. Advancing the state of mobile cloud computing. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, pages 21–28, New York, NY, USA. ACM.

Núria Bel. 2010. Platform for automatic, normalized annotation and cost-effective acquisition of language resources for human language technologies. panacea. *Procesamiento del Lenguaje Natural*, 45:327–328.

Frances Cairncross. 2001. *The death of distance: How the communications revolution is changing our lives*. Harvard Business Press.

Nancy Ide, James Pustejovsky, Christopher Cieri, Eric Nyberg, Denise DiPersio, Chunqi Shi, Keith Suderman, Marc Verhagen, Di Wang, and Jonathan Wright. 2014. The language application grid. *Proceedings of the Ninth International Language Resources and Evaluation (LREC2014)*, pages 22–30.

Toru Ishida, Yohei Murakami, Donghui Lin, Masahiro Tanaka, and Rieko Inaba. 2012. Language grid revisited: An infrastructure for intercultural collaboration. In Yves Demazeau, Jrg P. Mller, Juan M. Corchado Rodrguez, and Javier Bajo Prez, editors, *Advances on Practical Applications of Agents and Multi-Agent Systems*, volume 155 of *Advances in Intelligent and Soft Computing*, pages 1–16. Springer Berlin Heidelberg.

Toru Ishida. 2006. Language grid: An infrastructure for intercultural collaboration. In *IEEE/IPSJ Symposium on Applications and the Internet*, pages 96–100.

Toru Ishida. 2011. *The language grid: Service-oriented collective intelligence for language resource interoperability*. Springer Science & Business Media.

Atta ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, and Samee Ullah Khan. 2014. A survey of mobile cloud computing application models. *IEEECommunications Surveys Tutorials*, 16(1):393–413.

Young Yun Kim. 2000. *Becoming intercultural: An integrative theory of communication and cross-cultural adaptation*. Sage Publications.

Susan Y. McGorry. 2000. Measurement in a crosscultural environment: survey translation issues. *Qualitative Market Research: An International Journal*, 3(2):74–81.

Yohei Murakami, Donghui Lin, Masahiro Tanaka, Takao Nakaguchi, and Toru Ishida. 2010. Language service management with the language grid. In *The Proceedings of the International Conference on Language Resources and Evaluation (LREC2010)*, pages 3526–3531.

Yohei Murakami, Donghui Lin, Masahiro Tanaka, Takao Nakaguchi, and Toru Ishida. 2011. Service grid architecture. In *The Language Grid*, pages 19–34. Springer.

Ingrid Piller. 2011. *Intercultural communication: A critical introduction*. Edinburgh University Press.

Stelios Piperidis. 2012. The meta-share language resources sharing infrastructure: Principles, challenges, solutions. In *The Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC2012)*, pages 36–42.

Shahryar Shafique Qureshi, Toufeeq Ahmad, Khalid Rafique, and Shuja ul islam. 2011. Mobile cloud computing as future for mobile applications - implementation methods and challenging issues. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 467–471, Sept.

Ron Scollon, Suzanne Wong Scollon, and Rodney H Jones. 2011. *Intercultural communication: A discourse approach*. John Wiley & Sons.

Lee Sechrest, Todd L Fay, and SM Hafeez Zaidi. 1972. Problems of translation in cross-cultural research. *Journal of cross-cultural psychology*, 3(1):41–56.

Tamás Váradi, Steven Krauwer, Peter Wittenburg, Martin Wynne, and Kimmo Koskenniemi. 2008. Clarin: Common language resources and technology infrastructure. In *The Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC2008)*, pages 1244–1248.