Toward A Model of Service Interaction Enabler in Mobile Environment

Eddie Leung, Maria Indrawan, Sea Ling
Monash University
900 Dandenong Road, Caulfield East,
Victoria 3145, Australia
+61 3 99032535
{Eddie.Leung, Maria.Indrawan, Chris.Ling}@infotech.monash.edu.au

ABSTRACT

Proliferation of mobile devices has posed challenges in the development of interaction models among devices. These models should allow the devices involved in an interaction to detect and analyze possible communications or services to invoke before the actual interaction can take place. In this paper, we propose a model that allows analysis of possible interactions of devices. Our model provides a logical structure to construct and compare the essential elements that defines the capability of a device. These elements can then be examined in the proposed interaction possibility analysis to determine possible interactions through the invocation of services.

Categories and Subject Descriptors
D.1.m [**Programming Techniques**]: Miscellaneous

General Terms
Algorithms, Design

Keywords
Device Interaction Analysis, Mobile Service Discovery, Spatial Model, Location Awareness.

1. INTRODUCTION

Enabling context awareness to service-based computing is a challenging task, in particular in modeling the interactions and the dynamic nature of the interactions. Many current service-based computing is built on the notion of centralized service discovery. With the proliferation of mobile devices, it is envisaged that there will be a need to perform peer level discovery rather than relying on a central discovery process. In mobile environment, before a particular service can be discovered and executed, it is important to perform initial check on the device ability in executing the services due to the heterogeneity nature of the hardware and software of mobile devices. In this type of environment, the discovery process is more than just finding the semantically matched services. It is also important to find the right level of possible execution based on the device's hardware and software capabilities. Unlike service-based computing that serve business processes, mobile service-based computing may not have predefined business model or workflow to follow. Hence it is important to check all the possibilities of interaction in an ad-hoc manner. In this paper, we present a model of service enabler interactions in mobile environment. In proposing this model, we do not consider service strictly as web service that may be invoked based on WSDL and SOAP protocol. We adopt the more abstract definition of service which is a task that can be completed by interacting with another process or application of another device. In our context, service discovery considers the process of finding appropriate level of interactions possible for a given task. In other words, it is assumed that devices involved in the interaction are aware of each other's presence, achieved through the use of location context. The 'discovery' process is then performed to find the appropriate level of service interactions.

Context is defined as any information that can be used to characterize the situation of an entity, which can be a person, place or object that is relevant to the interaction between a user and an application [3]. In our project, context, in particular location proximity, is used as a trigger to initiate interaction among devices. It is essential because of the following reasons. Firstly, location information is an essential context for user to carry out tasks. Humans are spatially located creatures. In any point in time, one must be facing a direction, having certain objects in sight and within reach of certain objects. The way we manage the spatial information of objects is an instinct, integrated with the way we plan, think and behave [7]. Secondly, although it is possible to design applications where human experience does not play a role, the application designer and developer (also human) tends to translate their experience with the environment into the logic of a program. Thirdly, context-aware systems are inherently bounded to locations (e.g. sensor or communication range), and therefore they cannot completely take location out of consideration [1].

We review a number of spatially based device interactions model and systems in the next section before presenting our proposed model, system architecture and implementation in sections 3 through 5. Section 6 concludes the paper.

2. RELATED WORK

In 1993, Benford and Fahlén [2] introduced a spatial model of interaction which allows interaction among objects by using a number of key concepts including *Space*, *Objects*, *Aura*, *Medium*, *Nimbus*, *Focus*, *Awareness*, *Adapters* and *Boundaries*. The model utilizes location information to produce spatial information, which can be as general as identifying the presence of other devices or as specific as acquiring information on the proximity of one device to the others. This type of information can be very valuable to assist device in autonomous decision-making. However, although it has been broadly recognized that spatial information can enhance the interaction and collaboration of mobile devices, the research in this area is limited.

Realizing that there is insufficient attention on utilizing spatial relation information for mobile devices interaction and collaboration, *Relate* [5] was developed to fill the gap in this area. The system consists of a set of customer sensor devices called Relate Dongles, which can be connected to mobile devices through Universal Serial Bus (USB). Instead of tracking absolute location of individual device, the Relate dongles capture spatial information such as the position and orientation of a device as well as relationship information such as whether other dongles are moving away or approaching by measuring the relative distance and angular bearing of other dongles that are within a 2 meters range using ultrasound. The system provides users with an interface which can be used to visually detect and locate other mobile devices that have dongles attached. File exchange between devices is also possible when there is a wireless connection [5].

Although Relate has proven relative location among devices can be accurately computed without instrumentation of the environment, the value of absolute location information provided by location sensing systems should not be undermined. It should be noted that systems which operate without instrumentation of the environment also imply that additional device or computation power is required to perform tracking of other devices. When most of the mobile devices have limited processing power, the flexibility is gained at the cost of increasing the processing burden on these devices.

Another recent research project in the area is *Digital Aura* by Ferscha et al. in 2004 [4]. The project proposes a thought model intended for managing spontaneous interaction among mobile devices via the use of the similar concept of Aura in the spatial model of interaction [2]. A digital aura is built on wireless technologies, such as Bluetooth, Infrared Data Association (IrDA) or Radio Frequency Identification (RFID). The physical range of an aura is determined by the technology used. Interaction between devices becomes possible when the auras collide, which means the "signal" of one aura comes across with the "signal" of another aura. A self describing profile encoded in eXtensible Markup Language (XML) will then be exchanged spontaneously between the devices that own the auras. An analysis is then performed on the profiles to find “sufficient” mutual similarity, and communications on application level between the objects are triggered accordingly [4].

The Digital Aura project has provided a realistic implementation which uses spatial information as context. However, under observation we have found that the system has a number of drawbacks as pointed out by Kiang, Indrawan and Ling [6]. Firstly, as the aura size is limited by the wireless technology used, the flexibility of increasing/decreasing the size of an aura to vary the possible interaction range is reduced. Secondly, since the notion of aura is implemented at the technology level, only one aura at a time is possible as a device can only use the one medium proffered by the wireless technology. Thirdly, as spatial information is limited to the presence/absence of other devices rather than more detailed location information, the involved devices will not be able to decide whether the interaction should be initiated based on the environmental information. For example, it may be desirable for a device to determine whether a large file should be transfer at a particular distance as the strength of signal decreases as the distance increases. Lastly, spontaneous interaction may not be favourable at all time. Even though filters can be used to protect a device from undesirable information, they will still compromise the limited computational power of the devices.

In 2007, Kiang, Indrawan and Ling [6] refined the Spatial Model of Interaction [2] by introducing an *Interaction Initiation Model*. The main purpose is to govern the process in which interactions among objects are initiated. In the model, the concept of compatibility is defined by aura type, where auras with the same type are considered as compatible for interaction. Similar to the Spatial Model of Interaction, an interaction is initialized when a collision between the auras takes place. Different types of collisions between auras are formally defined by a set of collision models. Each collision model formally specifies what type of contact between auras should be considered as collision based on various level of strictness. Systems that utilize the model can then mix and match the various collision models according to the objectives.

Although Kiang’s work has eliminated some vague areas of the Spatial Model of Interaction [2], the model does have a major shortcoming. In Kiang’s model, aura type is one of the critical elements for determining whether an interaction is possible as collisions of auras of different types are disregarded. The model, however, does not provide any formal definition on what aura type is or how aura type is defined. Furthermore, as aura type is not properly defined there is no standard guideline on how various aura types are compared. As a result, the model is only capable of handling the most basic result, *exact match*, of a comparison. It fails to identify the relationships among aura types, which leads to an undesired result: interaction between devices immediately become impossible when aura type does not exactly match with each other. No alternative path can be considered even though there are some obvious relationships among many aura types. The relationships among many aura types are represented by the *device capability model* in our proposed framework.

3. DEVICE CAPABILITY MODEL

Based on our investigation, we have identified four major considerations in designing mobile service discovery and interactions: 1) Capability - Besides recognizing the interested service interactions, a device must be able to determine whether the other involved devices, or even itself, is capable of a particular service interaction. An example of a capability is video conference. In order to initiate a video conference between two devices, the foremost requirement is to ascertain that both involved devices can handle video conferencing. How can devices discover and determine their own, or even the other parties’ capabilities? What elements can be used to determine the capabilities of devices? 2) Capability compatibility - When the capabilities of devices are determined, they need to know whether these capabilities are compatible with each other. How can devices ascertain the compatibilities between various capabilities? 3) Capability Information Exchange – In order to determine the capabilities of each other, devices must exchange information about their capabilities at some point. When should this information exchange take place? 4) Interaction Degradation - Similar to most other objects in the world, there are some relationships between various types of capability. For example, there is possibly some relationship among video conference, audio conference and text-based conference. When a device is determined as incompatible for a particular capability, would interaction degradation become possible through these relationships?

To address the four issues, we propose a capability model along with an analytical approach called interaction possibility analysis. The proposed capability model aims to identify the basic elements that define the physical capabilities of a device. These elements can then be used in interaction possibility analysis to determine a possible interaction among these devices. The model also introduces location awareness to allow location information as the context information to determine when capability information exchange should take place. In addition, through analyzing the result of the interaction possibility analysis, the relationships among various interactions can be identified. Based on this relationship information, appropriate level of degradation can then be offered when interaction is considered as incompatible.

Benford and Fahlén [2] introduced a number of abstract concepts into the spatial model of interaction to govern interactions among objects. These abstract concepts are *Space*, *Objects*, *Aura*, *Medium*, *Nimbus*, *Focus*, *Awareness*, *Adapters* and *Boundaries*. To realize the model, Kiang, Indrawan and Ling [6] modified these concepts and developed an interaction model for use in the real environments. The proposed capability model is also developed based on these concepts with a number of new concepts and modification. The model provides a logical structure to construct devices. Devices that are constructed using the capability model can then be used in Interaction Possibility Analysis, which we will present in Section 4. The list below provides definitions for each the concepts of the model:

- ***Interaction*** - We define *Interaction* as an act in which two or more *Objects* communicate with, cooperate with, share information with, or affect one or each other through performing a mutual function at the same time. For instance, interaction between two *Objects* can be in a form of conducting a video conference, playing a device-device game, transferring a file, or providing remote system support. The feasibility of initiating an *Interaction* is determined by three elements: the proximity with other *Objects*, the physical ability of the involved *Objects* and the desire of an *Object* to participate an *Interaction*.

- ***Objects*** - *Objects* are the entities that are capable to initiate an *Interaction*. The positions of *Objects* are monitored by location tracking system, and based on this information, possible *Interactions* among *Objects* are identified. An *Object* must have certain level of processing power and appropriate networking device for wireless communication. Examples of *Object* are laptop, PDA and mobile phone

- ***Resource*** - It is defined as the basic unit for defining the *Capabilities* of an *Object*. It can be in the form of hardware or software. In order to initiate an *Interaction*, the involved *Objects* must have sufficient *Resources* to support the functions required in the interaction. Examples of *Resource* required for video conferencing are video camera, microphone, speaker, screen, keyboard (hardware), and video conference client (software). In order to conduct a video conference, an *Object* must be equipped with all these resources.

- ***Capability*** - It is formally defined as a logical grouping of *Resources* used for describing a specific physical capability of an *Object*. It determines whether an *Interaction* is possible from the physical ability point of view. A *Capability* is formed by all *Resources* required to support the functionalities of a specific interaction. The *Resources* must include both software and hardware. When *Resources* required for a *Capability are* equipped, an object is considered as capable of interacting with other objects by using the *Capability*. For instance, when an *Object* is equipped with the *Resources* screen, keyboard, appropriate networking devices and MSN messenger, the *Object* is considered as capable of interacting with other *Objects* with same *Capability* through chatting and transmitting file facilities, which are the functionalities of MSN messenger that can be performed with the support of screen, keyboard and networking devices. Although MSN messenger offers some other facilities, only those that have the *Resource* requirements fulfilled will be enabled. There are few points need to be noted:

  - The *Resource* set that forms a *Capability* is always unique within an *Object*. Hence, in the process of initiating an *Interaction,* one can determine the *Capability* required by identifying the *Resources* needed.

  - A *Capability* does not equal to a particular functionality. Instead, it represents the functionalities that can be initiated with the support of the *Resources* that forms the *Capability*.

- ***Awareness*** - *Awareness* (consciousness) is a concept that is used to determine whether an *Interaction* via a particular *Capability* is possible based on the level the *Object* desires to participate in an *Interaction*. It is manipulated by employing the concepts of *Focus* and *Nimbus*.

- ***Nimbus*** - The notion of *Nimbus* is used in cooperation with *Focus* to manipulate the *Awareness* of a *Capability* of an *Object*. It is represented as numerical value in the proposed model. When the *Nimbus* value of a *Capability* X of an *Object* A is higher than the *Focus* value of *Capability* X of another *Object* B, *Capability* X of *Object* A is concealed from *Object* B. As a result, *Object* B is not allowed to interact with Object A through *Capability* X. It should be noted that this is merely a simplified implementation of *Nimbus*. Additional levels of *Awareness* can be acquired by using various differences between the value of *Nimbus* and *Focus* of two *Objects*.

- ***Focus*** - The notion of *Focus* is used in cooperation with *Nimbus* to manipulate the *Awareness* of a *Capability* of an *Object*. Similar to *Nimbus*, *Focus* is represented as numerical value in the proposed model. When the *Focus* value of a *Capability* X of an *Object* A is higher than the *Nimbus* value of *Capability* X of another *Object* B, *Object* A is aware of the *Capability* X of *Object* B. Hence, *Object* A can interact with Object B through *Capability* X.

- ***Aura*** - The concept of *Aura* is closely akin to the one originally proposed in the initiation model of interaction of Kiang, Indrawan and Ling [6]. An *Aura* is a conceptual subspace that acts as an interaction enabler of one or multiple *Capabilities* offered by the same application. It can be switch on or off to activate/ deactivate interaction detection of the *Capability* it represents. An *Aura* determines whether an interaction is possible from the proximity point of view. *Objects* carry their *Auras* with them as they move around. When there is a collision between *Auras* of two *Objects*, an *Interaction Possibility Analysis* is performed to determine whether an *Interaction* can be initiated based on the *Capabilities* and *Awareness* of the two involved *Objects*. If the requirements for either *Capabilities* or *Awareness* are not met, no interaction will be initiated. In the other words, *Aura* provides a primary mechanism to determine if two *Objects* can interact based on their proximity with each other at the initial stage. The approach (mechanism) for connection establishment between objects is implementation specific. Implementer is responsible to specify the information on this.

Figure 1 provides a visualization for the abstract concepts above in a diagram. As shown in the figure, an *Object* may have multiple *Aura*s; each *Aura* represents one or more *Capabilities* offered by the same aura-enabled application. A *Capability* is formed by a number of *Resources* and a *Resource* may be shared by multiple *Capabilities*. It should be noted that *Capabilities* that share the same *Resources* can be any *Capabilities* of an *Object*. They are not necessarily offered by the same application.

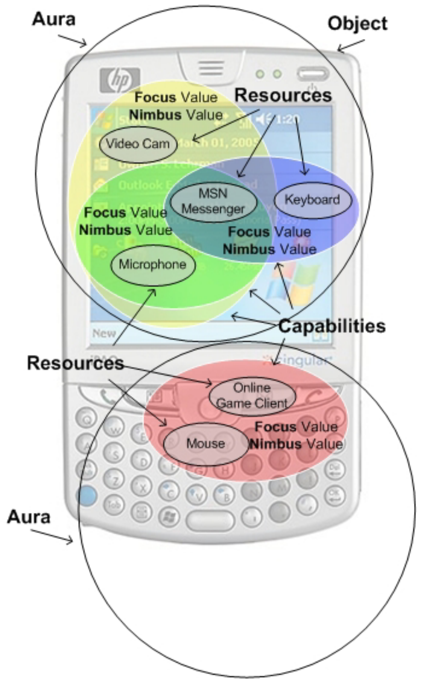


Figure 1 Visualization of Capability Model

4. INTERACTION POSSIBILITY ANALYSIS

The proposed capability model provides a logical structure to construct aura-enabled devices. By constructing devices using the capability model, the basic elements that define the physical capabilities of a device are identified. These elements can then be used in the proposed analytical approach, Interaction Possibility Analysis, to determine a possible interaction among these devices. Through the use of the capability model together with the proposed analytical approach, the decisions of whether interactions among devices are feasible will be made based on the actual capabilities of the involved devices, which is much more natural in comparison to some arbitrary representation. Moreover, as the most basic elements that define capabilities are compared during the analytical approach to determine the possibility of interaction, the differences of elements can be used to identify the relationships among various capabilities. By analyzing this relationship information, appropriate degradation can be offered when interaction is considered as incompatible.

The analysis is triggered when *Auras* of two *Objects* collide. It consists of two stages of checking: *Capability Compatibility Check* and *Awareness and Concealment check*. If the requirements of any stage are not met, the analysis will terminate and no further action will be taken.

4.1 Capability Compatibility Check

In Capability Compatibility Check, the *Capabilities* represented by the collided *Auras* will be examined to determine whether the *Objects* have certain common *Capabilities* that can be used to initiate an *Interaction*. The examination will be carried out on the *Resource* level, which is the basic unit that forms a *Capability*.

The analysis ends with one of the four possible scenarios: Exact Match, Subset, Intersection and No Match. The scenarios identify the relationships between the examined *Capabilities*. By analyzing the relationship, the possibility of an *Interaction* is determined. To define the four possible scenarios formally, the preliminary definition below will be used:

Preliminary Definition:

Given an object o, let $Capability_o$ be a capability of object o and r_i (where i is a natural number) be a member of a resource set that forms a capability. Formally, a capability of object o can be defined as:

$$Capability_o = \{r_1, \dots r_n\}$$

- **Exact Match Scenario** - When the two *Resource* sets that are compared exactly match with each other, an *Exact Match Scenario* occurs. Formally, the scenario is defined as:

  Definition 1:

  For any two objects x and y, exact match scenario occurs when:

  $$Capability_x = Capability_y$$

  Figure 2 provides a graphical representation of an exact match scenario.

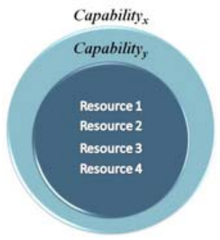


Figure 2 An *Exact Match Scenario*

When exact match scenario occurs, the two *Objects* are equipped with all resources required for that particular *Capability* and hence the interaction between the two *Objects* via the *Capability* is feasible from a physical ability point of view. In Figure 2, resources that form *Capability* of x and *Capability* of y equals to each other and therefore initiation of the interaction are physically possible.

- **Subset Scenario** - When one of the compared *Resource* set is a subset of another, a *Subset Scenario* occurs. The scenario is formally defined as:

  Definition 2:

  For any two objects x and y, subset scenario occurs when either:

- $Capability_y \subset Capability_x$; or
- $Capability_x \subset Capability_y$

Figure 3 depicts a subset scenario.

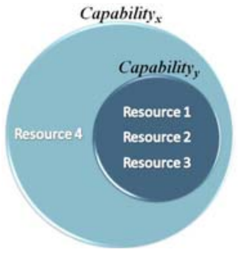


Figure 3 A *Subset Scenario*

It is often that an object with insufficient *Resources* to perform a particular operation can be capable of a similar task with lower requirement. For example, provided that the only difference between the *Resources* required for conducting a video-conference and audio-conference is a video camera. An *Object* can conduct audio-conference but without a video camera is not capable of a video conference, as a video camera is requisite to capture the motion. However, the *Object* can hold an audio-conference as long as the *Resources* it carries fulfill the requirements of audio-conference. In a similar sense, a device is not equipped with microphone and therefore is not competent for an audio conference can in fact support text-based conferencing by using appropriate input device. When a subset match scenario occurs, the super resource set must downgrade its resource requirement to match the sub resource set in order to initiate an interaction.

- **Intersection Scenario** - When there is only a partial match between both of the compared *Resource* sets, an *Intersection Scenario* occurs. The scenario is formally defined as:

  Definition 3:

  For any two objects x and y, intersection scenario occurs when the following conditions are satisfied:

  - $Capability_x \cap Capability_y \neq \varnothing$;
  - $Capability_y \not\subset Capability_x$; and
  - $Capability_x \not\subset Capability_y$

  Figure 4 depicts an intersection scenario.

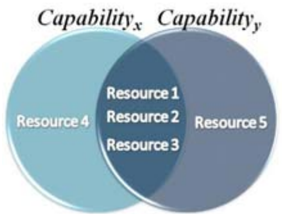


Figure 4 An *Intersection Scenario*

When intersection scenario occurs, the *Objects* need to search for a *Capability* that only requires resources that are common to both *Objects*. If the *Capability* is found, *Interaction* between the two *Objects* is possible.

- **No Match Scenario** - When there is no match between the compared resource sets, a *No Match Scenario* occurs. Formally, the scenario is defined as:

  Definition 4:

  For any two objects x and y, no match scenario occurs when $Capability_x \cap Capability_y = \varnothing$

  Figure 5 shows a no match scenario.

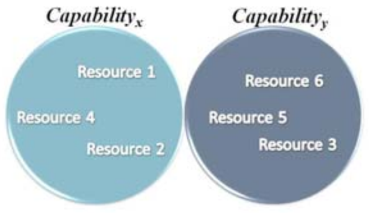


Figure 5 A *No Match Scenario*

The no match scenario indicates that the *Capabilities* the collided auras represent are different from each other and there is no relationship between the two *Capabilities*. Hence, no interaction can be initiated.

If at least one common *Capability* is found, the *Interaction* is considered as physically feasible and the analysis will move to the Awareness and Concealment Check. In contrast, if the requirements of *Resources* are not met, the analysis will terminate and no further analysis will be performed.

4.2 Awareness and Concealment Check

In Awareness and Concealment Check, the value of *Nimbus* and *Focus* pair of the common *Capabilities* represented by the collided *Aura* will be compared against each other. If a *Capability* of one Object can be aware by another, an *Interaction* is allowed to be initiated. Conversely, if the required *Capability* of both *Objects* is concealed from each other, the analysis ends and no action will be taken.

To illustrate how the *Nimbus* and *Focus* pair of the common *Capability* are examined in Awareness and Concealment Check, assume that there are two *Objects x* and *y* and the two *Objects* have a common *Capability*. If the value of *Focus* of the *Capability* of *Object x* is higher than the value of *Nimbus* of the *Capability* of *Object y*, *Object x* can be aware of the *Capability* of Object *y*. Thus, *Object x* can initiate an *Interaction* with *Object y* via the *Capability*. On the other hand, if the value of *Nimbus* of the *Capability* of *Object x* is higher than the value of *Nimbus* of the *Capability* of *Object y*, the *Capability* of *Object x* is concealed from *Object y*. As a result, *Object y* cannot initiate an Interaction with *Object* A through the *Capability*. It should be noted that, in the later example, although *Object y* cannot be aware of that particular *Capability* of *Object x*, *Object y* may still be aware of other *Capabilities* of *Object x* and therefore may still be able to interact with *Object x* via other *Capabilities*.

It should be noted that interaction possibility analysis only acts a guide line on how to analyze the result of comparison on objects constructed using capability model. The actual algorithm for comparing objects is implementation specific. In addition, the component which is responsible for performing the analysis and the location where the analysis is performed is not limited. The analysis does not even need to be carried out in a single component. The processes required can be separated into multiple remote components according to the needs of the implementer.

5. SYSTEM ARCHITECTURE AND IMPLEMENTATION

The proposed system architecture incorporates the capability model and interaction possibility analysis proposed in Section 3 and Section 4. The architecture is designed based on the architecture of Kiang, Indrawan and Ling [6] with a number of enhancements to accommodate the new model into the system. Figure 6 provides a high-level graphical representation of the proposed architecture.

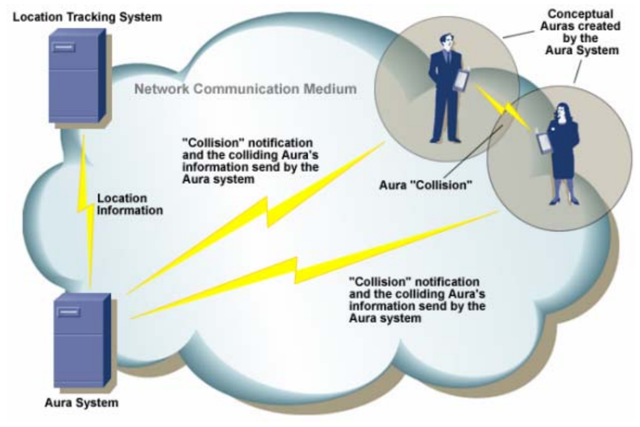

Location Tracking System
Conceptual Auras created by the Aura System
Network Communication Medium
"Collision" notification and the colliding Aura's information send by the Aura system
Location Information
Aura "Collision"
"Collision" notification and the colliding Aura's information send by the Aura system
Aura System

Figure 6 High Level View of System Architecture [6].

The Aura system in the figure queries the location tracking system for location information, which will be used by the Aura system to create conceptual auras around objects that are constructed using capability model and to monitor collisions among these auras. When a collision is detected, the Aura system will perform an interaction possibility analysis to determine whether there is any possible interaction. If a possible interaction is found, notification will be sent to the involved objects. The notification contains information about the colliding Aura and the common capabilities that can be used for interaction. The involved application can then interact with the counterpart of the colliding objects using this information. Figure 7 provides a more detailed view of the system architecture. As shown in the figure, there are four main components of the architecture: user, aura system, location tracking system and device

- **User** - They are the end-user of the system. In order to be a user of system, he/she must have an aura-enabled device.
- **Location Tracking System** - In the proposed architecture, user's devices utilize some form of location tracking mechanism which sends raw data to a location positioning system. The location positioning system will then transforms the received raw data into location coordinates.
- **Aura System** - Aura system monitors a list of aura-enabled applications that have provided information of their capabilities and the device they are installed on to the system. The aura system periodically queries the location tracking system for location information of those devices to calculate the relative distances for creating conceptual aura and detecting aura collision. When collision is detected, the aura system will perform an interaction possibility analysis to determine capabilities that can be used for interaction. Based on the result, notification will be sent in a form of event to the interaction manager of the involved devices.

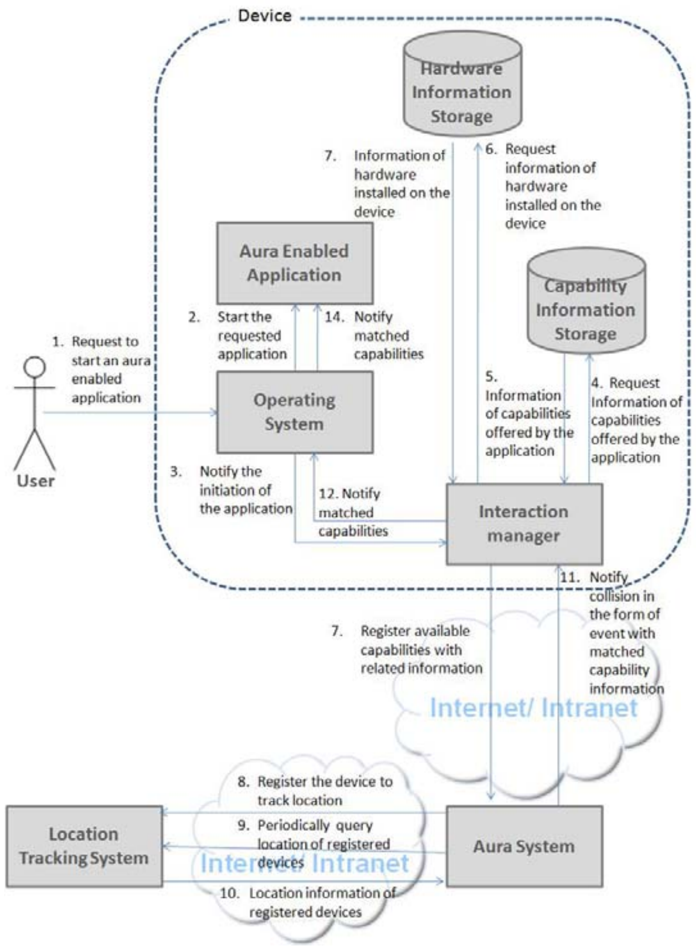

Device
Hardware Information Storage
7. Information of hardware installed on the device
6. Request information of hardware installed on the device
Aura Enabled Application
Capability Information Storage
2. Start the requested application
14. Notify matched capabilities
1. Request to start an aura enabled application
User
Operating System
5. Information of capabilities offered by the application
4. Request Information of capabilities offered by the application
3. Notify the initiation of the application
12. Notify matched capabilities
Interaction manager
11. Notify collision in the form of event with matched capability information
7. Register available capabilities with related information
Internet/ Intranet
8. Register the device to track location
9. Periodically query location of registered devices
Location Tracking System
Aura System
Internet/ Intranet
10. Location information of registered devices

Figure 7 Low Level View of System Architecture

- **Device** - The aura-enabled devices used by users and constitute *Objects* in the capability model. They can be in many forms from desktop PCs or smart refrigerator to laptops to handheld mobile devices such as PDAs and Tablet PCs. Fundamentally, any device with computational processing power and communication capability can be categorized as device in the architecture.

  As depicted in Figure 7, the sub components installed on the device include:

  - **Operating System** - A program that manages the resources of a device such as processor, memory, disk, networking activities and other applications. In the proposed architecture, it is responsible to notify interaction manager when any Aura-Enabled Application is initiated by user and manage the update of capability information storage and hardware information storage when there is any change in aura-enabled application and hardware.
  - **Aura-Enabled Application** - Applications that support the proposed capability model. When it is installed, it notifies operating system about all the capabilities it offers, the resources required to support these capabilities and the value of nimbus and focus of each capability. Operating system will then update the capability information storage based on the information.

- **Capability Information Storage** - A data storage that stores information about capabilities offered by all aura-enabled application, the resources required to support the capabilities and the value of nimbus and focus of each capability. It can be in many forms such as a relational database file, text file or XML file.
- **Hardware Information Storage** - A data storage that stores information about what hardware is available on the device. It can also be in any form such as a relational database file, text file or XML file.
- **Interaction Manager** - A program that acts as a bridge among aura-enabled application, operating system and aura system. Upon receiving notification from operating system when an aura-enabled application is initiated, it will determine what capabilities offered by this application are available based on the installed hardware information. The information of the available capabilities will then be sent to the aura system to create a conceptual aura for the capabilities. It also provides features that enable users to customize the attributes of individual capability such as nimbus and focus. It is also responsible for receiving collision notifications sent by the aura system and notify appropriate application based on the notification..

5.1 Considerations on Different Approach of Capability Relationship Definition

In Section 3, we have defined that a capability is created as a set of resources. There are two possible approaches in forming capabilities. One is to create a dynamic capability list; the other is to create a relatively static list.

In the dynamic approach, the capability list is created on-the-fly during the capability compatibility check in the interaction possibility analysis. This approach allows maximum flexibility, as devices can try to form a capability to match another being checked using any possible resources available in a case of subset or intersection scenario. However, these advantages do come with a cost at the efficiency as the capability formation is performed every time when a subset or intersection scenario occurs. The devices with capability defined in this approach must have high processing power in order to handle the search burden.

Another approach is to define the relations statically. This approach creates capabilities during the initial deployment of the aura system. The system defines all possible capabilities that can be recognized by the system. An update after the first deployment is possible. However, it will involve updates to be propagated to all devices in the system. In a case of subset or intersection scenario, the devices will only need to determine whether there is any predefined capabilities that can match the subset or intersection. Hence, the number of searches required during the analysis is significantly lower compared to the dynamic approach. In contrast to the processing burden of the dynamic approach, the static approach gains the efficiency at the cost of flexibility.

Based on the needs of implementers, different approaches can be used to fit the requirements of a particular system.

5.2 Considerations on Different Level of Aura Implementation

Depending on the requirements of the implementer, *Aura* can be implemented using different levels of granularity. The three possible levels are object level, application level and capability level. Object refers to the aura-enabled devices that are monitored by the location tracking system, whereas application refers to the computer program installed on the object, and capability refers to the functionality offered by the aura-enabled application.

If *Aura* is implemented at the object level, a device is represented by one *Aura* in the system. All aura-enabled applications on the device rely on the same aura to keep track of other aura-enabled devices in the surrounding area for possible interaction. If *Aura* is implemented at the application level, a device may have multiple *Aura*s depending on the number of aura-enabled applications installed. Each aura-enabled application installed on the device is represented by one *Aura*, which will be used by the application it represents to detect possible interaction. Finally, if *Aura* is implemented at the capability level, an *Aura* represents a single interaction capable functionality offered by an aura-enabled application on the device.

The choice of implementation level has a direct impact on the way how interaction possibility is performed due to the different level of information carried by an *Aura*. The following discusses the differences between different levels of *Aura* implementation.

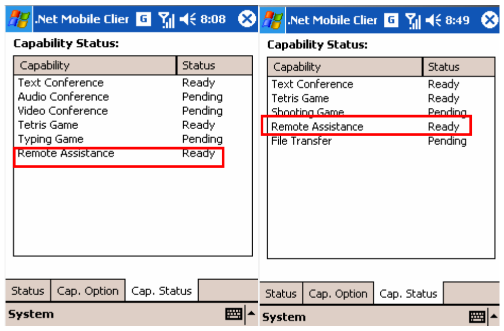
- **Capability Level** - When *Aura* is implemented at the capability level, where an *Aura* only carries the information of a particular capability it represents, part of interaction possibility analysis is forced to be switched from the aura server to the interaction manager of the involved device When a subset or an intersection scenario occurs, additional analysis is required to search for possible alternative capability for interaction, as the aura manager server does not have any information on what other available capabilities are in the involved devices. Although this approach may relieve some processing pressure of the aura manager server, serious performance issues may establish as the processing burden are switched to the devices, which normally have far less processing power compared to the server. Because of this reason, the capability level implementation may not be suitable for system in which most of the devices are handheld devices.
- **Object Level** - In contrast to *Aura* implemented at capability level, an *Aura* implemented at object level contains the information of all available capabilities offered by the device that the *Aura* represents. As a result, the interaction possibility analysis can entirely be processed in aura manager server.

  The process of exact match scenario at object level is identical to the counterpart at capability level. However, the processes of subset and intersection scenario at object level are significantly simplified. The reason is that an Aura implemented at object level contains information of all capabilities offered by a device, the aura manager server would therefore have sufficient information to process the entire interaction possibility analysis locally. As aura manager server usually has much higher computational power than the devices being tracked, the performance issues mentioned previously in the capability level implementation can be eliminated. Nevertheless, the level does have its drawback. As an *Aura* in the system carries the information of all capabilities offered by a device, the aura manager server may need to process a large number of unnecessary searches during the capability compatibility check to analyze every single capability offered in both *Auras*. As a result, a considerable amount of processing power is wasted.

- **Application Level** - The implementation at application level offers a solution to address the issues of implementations at both capability level and object level. Interaction possibility analysis involved in the application level is nearly identical to those involved in object level, except for the much smaller number of capability analysis required in each collision as each *Aura* in the system is restricted to represent only the capabilities offered by one application. As devices can only interact with compatible capability offered by the same application, even though the number of capabilities represented by an *Aura* may be reduced, the potential opportunity of interaction will still not be compromised

5.3 Implementation

Based on the system architecture, we have developed a prototype as a proof of concept. The prototype was developed using both C# 2.0 and Java. The inter-components communications are handled by socket technology and XML web service. However, it should be noted that the device capability model, interaction possibility analysis and the system architecture are not tied to any particular technology. Figure 8 shows a screen capture of the prototype.



Capability Status:

Capability	Status
Text Conference	Ready
Audio Conference	Pending
Video Conference	Pending
Tetris Game	Ready
Typing Game	Pending
Remote Assistance	Ready

Capability Status:

Capability	Status
Text Conference	Ready
Tetris Game	Ready
Shooting Game	Pending
Remote Assistance	Ready
File Transfer	Pending

Figure 8 A Screen Capture of the Prototype

6. CONCLUSION

Our paper has been concerned with context-based service interaction discovery and its execution in mobile environment. Specifically, we have introduced a model which provides a logical structure for constructing and comparing the essential elements that defines the capability of a device. The key points of our model are:

- Software and hardware installed on a device are the most basic elements that define the capabilities of a device.
- Aura is used as a service interaction enabler to initiate capability information exchange based on the proximity of devices
- Nimbus and Focus are used to determine the awareness and concealment level of a device.

We have also introduced an interaction possibility analysis which compares the devices constructed using our proposed model. Through analyzing four possible results, namely Exact Match, Subset, Intersection and No Match, the possibility of service interactions and relationships between the service interactions can be determined.

Finally, we have proposed a system architecture which incorporates the proposed capability model and interaction possibility analysis.

With this work established, further research is now required. Some possible research directions are:

- *Standard Interpretation of Resources*: The need for resource standardization is driven by the need of interoperability. To allow the model to be applied in an opened environment, the interpretation of resources must be consistent across different devices. Future research may consider using ontology to establish a standard for resources.
- *Multi Level of Object*: In the current model, there is only one level of object which represents a device of a user. To enhance the scalability, future researchers can extend object to multi level, so that an object can contain a number of sub-objects. For example, an object representing a smart lecture hall may consist of a number of lower level objects, such as projector, sound system and lighting system, which can all interact with other objects in the model.
- *Peer to Peer System Architecture*: Although the proposed model and interaction possibility analysis are not tied to any particular system architecture, the proposed system architecture is currently developed in a server-client manner. The architecture is suitable for organization that has a centralized server running at any time. To allow the model to be applied in a more general environment, future developer may consider switching the system architecture to a peer-to-peer architecture.

7. REFERENCES

[1] Beigl, M., Zimmer, T., et al., A Location Model for Communicating and Processing of Context. Personal Ubiquitous Comput. 6 (2002) 341-357.

[2] Benford, S., and Fahlén, L., A Spatial Model of Interaction in Large Virtual Environments, Third European Conference on CSCW (ECSCW'93), Milano, Italy, Kluwer, 1993, pp. 16.

[3] Dey, A. K., and Abowd, G. D., PANEL: Toward a Better Understanding of Context and Context-Awareness, GVU Technical Report, College of Computing, Georgia Institute of Technology, 1999.

[4] Ferscha, A., Hechinger, M., et al., Digital Aura, Pervasive Computing, Second International Conference, Pervasive 2004, Vienna, 2004.

[5] Gerd, K., Christian, K., et al., Sensing and visualizing spatial relations of mobile devices. Proceedings of the 18th annual ACM symposium on User interface software and technology (2005) 93-102.

[6] Kiang, V. J. K., Indrawan, M., Ling, S., Realising Aura for Initiating Interactions in Real Environments, Faculty of Information Technology and Systems, Monash University, Melbourne, 2006, pp. 136.

[7] Kirsh, D., The Intelligent Use of Space. Artificial Intelligence 73, Issue 1-2 (Feb 1995), 31-68.