# Mobile Middleware for Situation-Aware Service Discovery and Coordination

Stephen S. Yau and Dazhi Huang

Department of Computer Science and Engineering

Arizona State University

Tempe, Arizona, USA

**Abstract**

Situation-aware (SA) service discovery and coordination can enable the dynamic integration of mobile devices with various network infrastructures (NIs), and hence allow mobile users to utilize various resources in NIs anytime and anywhere. Various challenges due to the heterogeneous mobile environments, resource limitations of mobile devices and user mobility still need to be addressed in order to achieve SA service discovery and coordination. These challenges can be effectively addressed by developing mobile middleware that provides a set of components for efficient context management, situation analysis, service discovery and coordination. In this chapter, the background, requirements, design issues and enabling techniques for mobile middleware for SA service discovery and coordination will be discussed.

**Table of Contents**

## 1. MOTIVATION

Recent advances in embedded systems, microelectronics and wireless communication technologies have increased the flexibility of using mobile devices for various practical applications that improve the personal productivity of users. However, mobile devices are still resource poor in comparison with computing resources in network infrastructures (NIs), such as the Internet, Grid and enterprise computing environments. Such NIs usually consist of non-mobile computing resources to provide high-performance computing and communication capabilities. Although these NIs have become more flexible and interoperable by adopting Service-Oriented Architecture [1], which can enable rapid composition of distributed applications regardless of the programming languages and platforms used in developing and running different components of the applications, it is still very difficult for these NIs to provide the desired flexibility to individual users, especially to mobile users, due to their immobility and large size. Hence, dynamic integration of mobile devices with NIs [2] is a subject in ubiquitous computing that has attracted much attention. Dynamic integration is the process with which a mobile device can detect, communicate with, and use the required services in nearby NIs in an application-transparent way. The benefit of dynamic integration is that the applications in both a mobile devices and an NI can interoperate with each other as if a mobile device itself is an integral part of the NI or vice versa [2]. The dynamic integration of mobile devices and NIs has brought a number of research issues, such as wireless ad hoc communication, service discovery and coordination, and distributed trust management. In this chapter, our discussion will be focused on techniques for service discovery and coordination for the dynamic integration of mobile devices with service-based NIs. Other topics are covered in the other chapters of this book.

In the service-based NIs, various capabilities, such as storage, computation and communication, are provided by different organizations as services and interconnected by various types of networks. We consider a service as a software/hardware entity with well-defined interfaces to provide certain capability over wired or wireless networks using standard protocols, such as TCP/IP, HTTP, and SOAP (Simple Object Access Protocol). By dynamically and seamlessly integrating mobile devices with such service-based NIs, users can utilize various capabilities in the NIs anytime and anywhere through their mobile devices. Moreover, the services in the NIs can be integrated following specific workflows, which are series of cooperating and coordinated activities designed to achieve complicated mission goals for users. Service discovery, which is the process of locating services that can satisfy the needs of

users, is a prerequisite of accessing the capabilities provided by the NIs. Service coordination is required to ensure the correctness of workflow execution. Service coordination is a process of monitoring the status of participant services, invoking proper participant services, and propagating necessary information to participant services to ensure the correct results obtained from the coordinated participant services.

To enable the effective integration of mobile devices with the service-based NIs, service discovery and coordination need to be situation-aware (SA) because of the following reasons: (1) Services may become unavailable or cannot provide desirable QoS due to distributed denial-of-service attacks, system failures or system overload. (2) Workflows may need to be adapted when the situation changes in order to achieve the users' mission goals. (3) New workflows may be generated in runtime to fulfill users' new mission goals. We consider *situation-awareness (SAW)* as the capability of being aware of situations and adapting the system's behavior based on situation changes [3, 4], which is needed for checking whether a service can meet the users' requirements and should be invoked and adapting workflows. A *situation* is a set of contexts in a mobile application over a period of time that affects future system behavior [3, 4]. A *context* is any instantaneous, detectable, and relevant property of the environment, the system, or users, such as time, location, light intensity, wind velocity, temperature, noise level, available bandwidth, and a user's schedule [3, 4].

However, achieving SA service discovery and coordination in mobile computing environments is challenging due to the heterogeneous environments, the resource limitations of mobile devices and the user mobility. Furthermore, developing mobile applications that make use of SA service discovery and coordination is difficult without appropriate system support. These challenges can be effectively addressed by developing mobile middleware that provides a set of components that can perform context management, situation analysis, service discovery and coordination efficiently. In this chapter, we will first review the background of mobile middleware for situation-aware service discovery and coordination. Then we will discuss the requirements, design issues and enabling techniques for mobile middleware for SA service discovery and coordination.

## 2. BACKGROUND

In this section, we will review the background of our topic from the following four aspects: situation-awareness, service discovery, service coordination and mobile middleware.

### 2.1. Situation-Awareness (SAW)

In this section, we will give an introduction to the literature on SAW and its related areas.

Early work on situation which was done primarily in artificial intelligence community focused on formalizing and reasoning on situations. Situation Calculus and its extensions [5-8] were developed for describing and reasoning how actions and other events affect the world, assuming that all actions and events changing the world are known or predictable. In Situation Calculus, a situation is considered as a complete state of the world, which leads to the well-known Frame Problem and Ramification Problem [6]. In [9], an opposite view of situation was considered, which formally defined a situation as a part of "the way the world M happens to be", and a situation supports the truth of a sentence Φ in M. In [9], Barwise defined "scene" as "visually perceived situation", and observed that a scene that people perceive consists of not only objects and individual properties associated with the objects, but also relationships between any two objects. Based on the concept of "scene", Barwise introduced *Situation Semantics* in [10], in which basic properties, relations, and situations are defined as objects [10]. Barwise's definition of situation is more practical compared to the definition of situation in Situation Calculus since Barwise's definition of situation allows the precise description of situations, and can be easily supported by the prevailing object-oriented modeling techniques. Currently, many other researchers have adopted Barwise's definition of situation, and developed their own formalisms of situations for various purposes, such as supporting SA software development, effective human-computer interactions and data fusion [3, 11-13]. For example, in [11, 12], a core SAW ontology was introduced based on a similar view of situations as Barwise's, which defines a situation as a collection of situation objects, including objects and relations as well as other situations.

Since early 90's, much research has been done by mobile computing community on context-aware computing. "Context" here usually refers to the information that can be used to characterize the situations on users, applications and environments although various researchers have used slightly different definitions [3, 4, 14-18]. Hence, context-awareness is considered as a part of SAW. Although a number of issues related to context-aware (or context-sensitive) computing have been discussed in [19-22], the term "context-aware computing" was first introduced in [14]. Since then, several frameworks, toolkits and infrastructures have been developed for providing support to context-aware application development. Notable results include CALAIS [23], Context Toolkit [17], CoolTown [24], MobiPADS [25], GAIA [26, 27], TSpaces [28] and Reconfigurable Context-Sensitive Middleware [3, 29]. CALAIS [23] focuses on applications accessible from mobile devices, and supports acquisition of contexts of users

and devices, but it is difficult to evolve existing applications when requirements for context acquisition and the capabilities and availabilities of sensors change. Context Toolkit [17] provides architectural support for context-aware applications, but it does not provide analysis of complex situations. CoolTown [24] supports applications that display contexts and services to end-users. MobiPADS [25] is a reflective middleware designed to support dynamic adaptation of context-aware services based on which application's runtime reconfiguration is achieved. GAIA [26, 27] provides context service, space repository, security service and other QoS for managing and interacting with active spaces. TSpaces [28] utilizes tuple spaces to store contexts and allows tuple space sharing for application software to read and write, but it ignores the status of the device where the application software executes, network conditions, and the surrounding environment as part of the overall context. Reconfigurable Context-Sensitive Middleware (RCSM) [3, 29] provides development and runtime support for SA application software, including a declarative Situation-Aware Interface Definition Language (SA-IDL) and its compiler for automated code generation, a reconfigurable SA Processor supporting runtime situation analysis and triggering proper actions of SA applications, and a RCSM Object Request Broker (R-ORB) supporting context discovery, acquisition and SA inter-object communications. Readers interested in context-aware computing are referred to [30, 31].

### 2.2. Service Discovery and Coordination

*Service discovery*, also known as *service location* or *matchmaking*, is the process of locating suitable services that can meet the users' requirements. During the past decade, substantial research has been done on service discovery, which generates various service discovery protocols, such as Jini [32], Salutation [33], Service Location Protocol (SLP) [34], Universal Plug-and-Play (UPnP) [35], and UDDI [36]. Currently, the major concerns on service discovery are: (1) How to identify the most suitable services that meet the users' needs? (2) How to utilize resources (network bandwidth, battery power, etc.), especially in mobile computing environments, efficiently in service discovery?

For (1), most existing service discovery approaches are based on syntactical matching, i.e. keyword or table based matching [37], such as UDDI. However, matching keywords or service interfaces are not good enough in many real-world scenarios to find suitable services that meet the users' needs. The semantics of services, the goals of users, the situations of users, systems and environments, the Quality-of-Service (QoS) that can be provided by the services, and the security requirements for using the services need to be considered in identifying the most suitable services. In order to understand the semantics of services, various models and languages [38-40] were proposed to

capture the service semantics. The goals of users are usually described using the corresponding query languages of the service description. Based on these models and languages, techniques for semantic-based service discovery were introduced [41-46]. Recently, security and other QoS have been incorporated in service discovery [47-49].

It has been shown that incorporating context-/situation-awareness in service discovery can greatly improve the precision and recall of the discovery results [45, 46, 50, 51], and hence improve the efficiency of mobile applications and reduce the distractions to users. For service discovery, *recall* is defined as the number of relevant services retrieved in service discovery divided by the total number of relevant services available, and *precision* is defined as the number of relevant services retrieved in service discovery divided by the total number of services discovered [45, 46]. In [50], contextual information, such as time, location, user name and device type, is used to differentiate services belonging to the same service category to increase the precision of service discovery. In addition, context history is analyzed to find useful patterns for predicting future service availability [50], which further increases the precision of service discovery. In [51], situation information is used to guide the generation of user profiles, and select appropriate user profiles for personalized information retrieval. Users' information retrieval requests are expanded based on their profiles and situations, which capture some implicit, but important characteristics of the information that the users want to retrieve, and hence improve the precision and recall of information retrieval. Although this work [51] is mainly for information retrieval, the idea can easily be applied to service discovery. In [45, 46], contextual information is used in two aspects: (a) It is used to make service requests information-richer to retrieve more relevant services, and hence increase the precision of service discovery, and (b) it is used to complete the contextual inputs required by relevant services when some of these inputs are not provided by service requestors to allow the retrieval of relevant services with missing inputs, and hence increase the recall of service discovery. However, so far there is no unified service discovery approach that considers service semantics, security policies, other QoS properties and SAW in the discovery process. Other noteworthy work on context-/situation-aware service discovery includes the discovery mechanisms in several context-/situation-aware platforms and middleware, such as Cooltown [24, 52], Context Toolkit [17] and RCSM [2, 53]. Although much progress has been made on SA service discovery, it is still a relatively new area, and much work needs to be done. In Section 4 of this chapter, we will discuss the design issues and enabling techniques for mobile middleware for SA service discovery.

For (2), since centralized service directories and/or registries are not always available in mobile computing environments lack of infrastructure support, service discovery in such environments often has similar characteristics to P2P (peer-to-peer) service discovery [54-57]. Given very limited communication bandwidth and battery power of mobile devices, the efficiency of service discovery protocols in mobile computing environments is of special interest to researchers in this area. Much research effort has been made on efficient service discovery in mobile computing environments [58-63]. We will not discuss this here since this is not in the scope of this chapter.

There are some industrial standards on service coordination, such as WS-Coordination [64] and WS-CF [65], and some notable approaches to context-aware service coordination [18, 66-68]. The industrial standards in [64, 65] aim at providing standard and extensible coordination frameworks to support coordinated workflows and transactions on web services, but do not address SAW in service coordination. In [18], a formal specification framework for modeling dynamically changing contexts and rules in contextual reactive systems for coordinating distributed systems was developed. MARS [67] aims at promoting context dependent coordination by incorporating the concept of programmable coordination media in distributed systems. Using "coordination contracts" to support the construction and evolution of complex service coordination was addressed in [66]. In EgoSpaces [68], a coordination model was introduced to focus on the context of a particular component in a mobile ad hoc network, and provide a middleware for context specification and runtime reconfiguration. In [18, 66-68], the service coordination is only based on the current context information. However, the changes of contexts over a period of time are also useful information, and should not be neglected. In Section 4 of this chapter, we will discuss the design issues and enabling techniques for mobile middleware for SA service coordination.

### 2.3. Mobile Middleware

Existing mobile middleware can be divided into two major categories depending on how they support the coordination among mobile devices: (1) tuple-space based, and (2) message based. Notable work in the first category includes LIME [69], TSpaces [28] and Limone [70]. Their tuple-space based coordination model supports location transparency and disconnected operations, and mobility is viewed as transparent changes in the content of the tuple space. Hence, they can easily support interactions among mobile devices. For tuple-space based mobile middleware, the major concerns are the scalability and performance of this type of middleware. Notable work from the second category includes ALICE [71], Mobiware [72], GAIA [26, 27], RCSM [3, 4], and MobiPADS [25]. ALICE [71] adds a mobility layer between the transport and the CORBA IIOP layers of the CORBA architecture to

support both mobile clients and servers. Mobiware [72] provides the facilities for managing an open, active, and adaptive mobile network by utilizing a CORBA-based architecture and using different adaptive algorithms as Java objects, which can be injected dynamically into mobile devices, access points, and mobile-capable network switches or routers. In the following subsections (Sections 2.3.1-2), we will give a brief overview to two mobile middleware, RCSM and MobiPADS, which provide context-/situation-awareness. Although there are other mobile middleware that can provide context-/situation-awareness, such as GAIA [26, 27], we will not introduce all of them here due to space limitation. Readers can refer to the other chapters of this book for information of other issues on mobile middleware.

### 2.3.1. RCSM

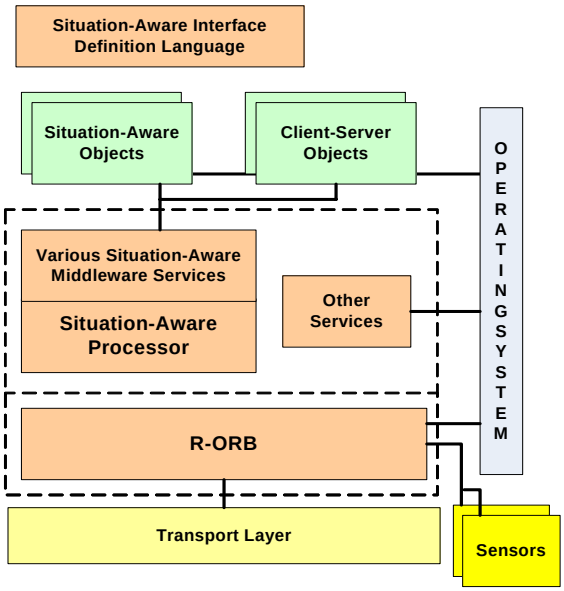


**Figure 1**. RCSM's architecture.

Reconfigurable Context-Sensitive Middleware (RCSM) is a lightweight SA middleware, which provides development and runtime support for SAW, dynamic service discovery and group communication for ubiquitous computing applications [2-4, 29]. A conceptual architecture of RCSM is shown in Figure 1. RCSM consists of the following major components:

1) SA Processor provides the runtime services for situation analysis and manages the SAW requirements of SA objects. The SAW requirements of SA objects are defined using Situation-Aware Interface Definition Language (SA-IDL) [3, 29]. An SA-IDL compiler has been developed to generate the situation-aware object skeleton codes and corresponding configuration files, which will be used by the SA Processor to perform situation analysis accordingly. The SA object skeleton codes provide the standard interfaces for SA objects to interact with the SA Processor.

2) RCSM Object Request Broker (R-ORB) provides the runtime services for context discovery and acquisition, and SA communication management. The Context Manager in R-ORB implements an efficient context discovery protocol [73] to support adaptive context discovery and acquisition in ubiquitous computing environments based on the requirements on contexts extracted from the configuration files of SA applications by the SA Processor. SA

object discovery protocols have also been developed to enable efficient and spontaneous communication between distributed SA objects [29, 53].

A unique feature provided by RCSM is the support for SAW. Using SA-IDL, contexts can be precisely described as context objects, and situations can be composed by not only the current values of multiple contexts, but also the historical values of multiple contexts over a period of time. The SA Processor is designed to cache and analyze the context history to determine the situation. In addition, the SAW requirements, such as the definitions of situations, can be modified in runtime through the SA Processor. Once the requirements have been changed, the R-ORB and SA Processor will reconfigure themselves to collect necessary contexts and perform situation analysis based on the new requirements.

**2.3.2. MobiPADS**

Mobile Platform for Actively Deployable Service (MobiPADS) [25] is a reflective middleware, which serves as an execution platform for context-aware mobile computing. MobiPADS enables active service deployment and reconfiguration in response to context changes, and hence it can optimize the operation of mobile applications when the operating context changes [25].

MobiPADS consists of two types of agents: MobiPADS server agents and MobiPADS client agents. MobiPADS server agents reside in the network infrastructure and are responsible for most of the optimization computations. MobiPADS client agents reside in the mobile devices and provide various services for mobile applications [25]. MobiPADS adopts the idea of mobile codes, and stores the codes of service objects in MobiPADS agents. Service objects can be deployed on either the client or server agent, and can migrate between the client and server agent when necessary (e.g., when the device where the client agent resides moves), which enable flexible reconfiguration of mobile applications. Each MobiPADS agent also has a set of system components for managing system configurations (MobiPADS client and server, and service objects), migrating service objects between MobiPADS server and client, recording known services, sending contextual event notification, and establishing virtual communication channels between service objects [25]. Each MobiPADS service is a pair of mobilets [25], which consists of a slave mobilet at the server agent for providing actual processing capabilities, and a master mobilet at the client agent for instructing the slave mobilets and presenting results to the client. In MobiPADS, the mobilets can be chained together to support necessary service composition for mobile applications, which is similar to

workflows in workflow systems. An XML-based language has been developed in MobiPADS to describe how service objects interact with each other and how they are configured [25].

MobiPADS utilizes an event subscription-notification model to provide context-awareness [25]. The idea is similar to the ECA (event-condition-action) model in active databases. All contexts are modeled as event sources, which will generate contextual events when certain conditions are satisfied. In MobiPADS, all the entities (system components, mobilets and mobile applications) can subscribe to contextual events of interests, and will be notified when certain events occur, and hence achieve context-awareness. MobiPADS also supports event composition [25], which allows combining multiple events from different context sources together to express complex semantics. However, MobiPADS only focuses on the current events [25], and does not consider historical events, which are important for achieving SAW.

## 3. REQUIREMENTS FOR MOBILE MIDDLEWARE FOR SA SERVICE DISCOVERY AND COORDINATION

To present a mobile middleware for SA service discovery and coordination to enable the dynamic integration of mobile devices with service-based NIs, it is necessary to first identify the requirements for mobile middleware for SA service discovery and coordination. The following is a list of requirements for such a middleware:

(1) **The capability to achieve SAW**. To achieve SA service discovery and coordination, the mobile middleware must be situation-aware. In [27, 29], several requirements for middleware for context-/situation-awareness are identified. Although the term “context-awareness” is used in [27], their “context” has a broader sense than others defining context-awareness, and is close to our “situation-awareness” [3, 4]. Hence, we also consider the requirements identified in [27] as the requirements for mobile middleware for situation-awareness. The following is a summary of the requirements for mobile middleware for situation-awareness identified in [27, 29]:

(R1.1) Support for specifying SAW requirements for mobile applications/agents, including the contexts and situations of interests, and applications’/agents’ behaviors in different situations.

(R1.2) Support for discovering contexts from the ambient environments based on the needs of various mobile applications/agents, acquiring contexts from various sources, and delivering acquired context data to mobile applications/agents.

(R1.3) Support for analyzing the acquired context data to determine the situation, and delivering the results of situation analysis to mobile applications/agents in a timely manner to trigger proper actions or adaptations.

(R1.4) Support for sharing situation and context information among different mobile applications/agents.

(R1.5) Support for runtime reconfiguration due to changes in SAW requirements of mobile applications/agents.

(R1.6) Support for incorporating various reasoning/learning mechanisms for situation analysis.

(R1.7) Syntactic and semantic interoperability among different mobile applications/agents.

Among these seven requirements for achieving situation-awareness, (R1.1 – R1.3) are the three basic requirements that a mobile middleware must satisfy for achieving situation-awareness. (R1.4 – R1.7) are desirable for better performance, flexibility, interoperability and extensibility.

(2) **The capability to accurately and efficiently locate necessary services to satisfy the users' needs.** The purpose of service discovery is to locate services that can satisfy the needs of users. The needs of users are determined by the users' goals. Whether a service is necessary for achieving a specific goal of users depends on the service semantics, such as what the service can do and how the service works. In a dynamic environment like mobile computing environments, both the users' goals and the service semantics may depend on the situations, i.e. the users' goals may vary and the services may have different behaviors when the situation changes. Also, the users may have certain preferences for different choices of services in different situations. Therefore, the mobile middleware needs to provide the following support for SA service discovery:

(R2.1) Support for developers or system administrators to specify service semantics precisely, especially how a service behaves when the situation changes.

(R2.2) Support for users to specify their goals and preferences in different situations.

(R2.3) Support for efficiently and accurately matching the service semantics and users' goals to locate the necessary services based on situations.

(3) **The capability to adaptively coordinate the execution of workflows consisting of multiple services to achieve the users' goals.** Sometimes a goal of users' cannot be achieved by invoking any service alone, but can be achieved by invoking multiple services in a coordinated manner. This is usually referred to as "workflow planning" or "service composition". "Workflow planning" or "service composition" is not the scope of this chapter, and hence will not be discussed here. Readers interested in this topic are referred to other chapters in this book. Here, we simply assume that a workflow planner [74, 75] is available and can

communicate with the mobile middleware. The workflows generated by existing workflow planners [74, 75] are usually static, i.e. it is assumed that the workflow planners have complete knowledge of the planning domain and can precisely (usually by reasoning) determine the status of the entire system in each step of the workflow execution. However, in real-world applications, this assumption is prone to be false, and hence SA service coordination is needed to ensure the correct execution of workflows. Since workflows usually consist of distributed services, and the execution of each service may have certain situational constraints (e.g. a service can be only be invoked in certain situations, or a service may have different behaviors when the situation changes), the following support needs to be provided by the mobile middleware for SA service coordination:

(R3.1) Support for analyzing a workflow to identify all the participant services in service coordination.

(R3.2) Support for invoking appropriate participant services based on the workflow and the situation, monitoring the status of participant services, and reporting the status of participant services to users or mobile applications/agents.

(R3.3) Support for detecting failures (e.g. violations of situational constraints on the execution of participant services, or unavailability of participant services) in workflow execution, identifying alternative services that can be used, and reporting the failures and current situation to the workflow planner when no alternative services can be found.

# 4. DESIGN ISSUES AND ENABLING TECHNIQUES FOR MOBILE MIDDLEWARE TO ACHIEVE SA SERVICE DISCOVERY AND COORDINATION

In this section, we will discuss the design issues and enabling techniques for mobile middleware to achieve SA service discovery and coordination from the following two aspects:

(1) Context management and situation analysis for achieving situation-awareness

(2) Incorporating situation-awareness in service discovery and coordination in mobile middleware

## 4.1. Context management and situation analysis for achieving situation-awareness

As discussed in Section 3, mobile middleware for SA service discovery and coordination must have the capability to achieve situation-awareness. Although existing mobile middleware, such as RCSM [2-4, 29, 53, 73],

MobiPADS [25] and GAIA [26, 27] has different approaches to achieving context-/situation-awareness, three important aspects for designing such middleware can be identified as follows:

(1) ***Modeling and specifying SAW requirements of mobile applications/agents***. To satisfy **R1.1, R1.6 and R1.7**, suitable models and languages for SAW requirements of mobile applications/agents must be developed. In [3], a model for SAW is presented. Based on this model, an interface definition language, SA-IDL [3, 29] was developed to allow developers to specify interfaces of SA objects and support the automated code generation of SA object skeletons. Each SA application is considered as a set of SA objects [3, 4], which will take various actions in various situations. These actions to be taken in various situations are abstracted as various functions of the SA objects [3, 4]. SA-IDL supports the specification and inheritance of context classes, and allows specifying the frequency of context acquisition [29]. For specifying situations, SA-IDL provides operators for context preprocessing and logical connectives for composing complex situations with various contexts within certain time range [3, 29]. Compilation of SA-IDL specifications will generate SA object skeletons, which can be extended by application developers by adding the implementations of functions to be triggered, and SA files for configuring RCSM [29]. In [25], context-awareness requirements are expressed by adaptive policies describing service composition and reconfiguration upon the detection of contextual events. XML is used to maintain the system profile, which contains adaptive policies for services and applications [25]. In [27], a predicate model of contexts is presented. This model can support the use of different reasoning mechanisms, such as first-order logic and temporal logic by agents to reason about contexts and determine their behaviors in different contexts. In [27], ontologies are used to describe context predicates for semantic interoperability. There are other models and languages for SAW (see Section 2). However, several important issues have not been addressed in modeling and specifying SAW requirements: (i) Incorporation of model/language primitives for representing spatial properties in SAW to support mobility, (ii) analysis of the expressiveness of various SAW models and specification languages, (iii) verification of SAW requirement specifications. Further investigation for these issues may utilize the results from existing research on ambient logic [76] and model checking [77].

- ***Context management***. To satisfy **R1.2 and R1.5**, it is necessary to develop components or services in mobile middleware to manage various sensing units (context sources) available on the mobile device, dynamically discover remote sensing units connected to other mobile devices in the neighborhood, and acquire and

propagate context data from local or remote sensing units to mobile applications. In MobiPADS [25], an *Event Register* has been developed, which enables application objects to subscribe to contextual event sources. Objects will be notified when the subscribed events occur. In GAIA [27], a Context Provider Lookup Service has been developed to allow context providers to advertise the contexts they provide, and support software agents to search for context providers that have the needed contextual information. In RCSM [29, 73], a component, Context Manager in R-ORB has been developed to manage local sensing units, acquire necessary contexts for applications, and perform context discovery if necessary. Since context discovery and context acquisition from remote sensing units require communication among mobile devices, which is likely to cause long delay and consume much energy, a key issue for developing such components or services in mobile middleware is the development of an adaptive, lightweight and energy-efficient protocol to perform context discovery and remote context acquisition in a timely manner. In MobiPADS and GAIA, this issue has not been addressed. In [73], a context discovery protocol, R-CDP is developed to address this issue. R-CDP combines pull and push communication paradigms for dynamic context discovery and efficient context retrieval. In R-CDP, context requesters dynamically discover contexts from context providers using pull communication paradigm, and the context providers proactively push updated context values to the requesters whenever the contexts undergo measurable variations. To improve energy-efficiency, R-CDP is designed to be aware of network conditions, and adaptively changes the interval of context request advertisements to alleviate network congestions, thus reduce retransmissions of context requests. R-CDP utilizes transmission probabilities to reduce redundant context result transmissions when there are multiple context providers in the network, and adaptively changes transmission probabilities as the number of context providers in the network varies. R-CDP also adopts and extends the Refresh Priority (RP) function [78] to determine when context providers should send updates of contexts, by calculating the RP based on the divergence of contexts (the difference between the previous and current context values) and energy consumption of context providers. However, R-CDP matches contexts using keywords in context advertisements, which cannot provide semantic interoperability, i.e. when different providers have different interpretations on the same keyword, or use different keywords for the same context. Ontology-based approaches, such as ontologies in GAIA for describing contextual information [27], could be combined with context discovery protocols like R-CDP to solve this problem.

- ***Situation Analysis***. To satisfy **R1.3**, **R1.4** and **R1.5**, it is necessary to develop components or services in mobile middleware to manage the SAW requirements of mobile applications/agents, maintain the history of context and situation of interests to mobile applications/agents, analyze relevant context and situation history to determine the current situation based on SAW requirements, and notify applications/agents when a situation of interest to them changes. A commonly used approach to situation analysis is to define situations in the form of logical rules, and to use a rule engine to infer situations in runtime. For example, in RCSM, situations and the actions to be triggered in different situations are defined by SA-IDL rules [29], and the SA Processor in RCSM is a rule engine that can process SA-IDL rules to analyze situations and determine what actions should be triggered [29]. The advantage of using a rule engine for situation analysis is that whenever a rule is changed, the rule engine will automatically perform reasoning with the new rule, which allow SAW requirements to be changed in runtime. However, such an approach also has a disadvantage that it is difficult for normal users to define complicated rules without making mistakes. Furthermore, in a rule-based approach, situations are inferred based on pre-defined rules, which cannot be changed without human interference. In [27], context synthesizers that can deduce higher-level contexts (our situations) using machine learning techniques like Bayesian networks are developed and used with context synthesizers based on rules. Combining learning with rule-based approach may overcome the abovementioned disadvantage, but also faces serious performance problem since machine learning techniques are computational-intensive, which is not suitable for resource-poor mobile devices. Further investigation for efficient on-line learning techniques or proper architecture that can off-load the computational intensive learning task from the mobile devices is needed.

### 4.2. Incorporating situation-awareness in service discovery and coordination in mobile middleware

Service discovery and coordination are generally provided as middleware services, such as the naming service and object transaction service in CORBA to ease the burden of service and application developers. As discussed in Section 2.2, the major advantages of incorporating SAW in service discovery and coordination, compared with traditional non-SAW service discovery and coordination techniques, are listed as follows:

a) It can greatly improve the ***precision*** and ***recall*** of discovery results [45, 46, 50, 51],

b) It enables adaptable service coordination [13, 25], which allows efficient and reliable workflow execution in unpredictable and dynamic environments.

Because of these advantages, SA service discovery and coordination can greatly improve the capability of mobile applications/agents to utilize various services in service-based network infrastructure effectively, and hence improve the performance and robustness of mobile applications/agents. Therefore, mobile middleware should provide services for SA service discovery and coordination. In this section, we will discuss the issues and enabling techniques for incorporating SAW in service discovery and coordination in mobile middleware.

❖ **Incorporating SAW in service discovery in mobile middleware**

Although various middleware may utilize different service discovery mechanisms, service discovery generally works in the following three phases [46]: (i) service request advertising, (ii) matchmaking, i.e. matching service semantics with service requests, and (iii) discovery result delivery. There are many useful ways of utilizing situation information in service discovery:

- Situation information can be used to expand service requests to provide more relevant information that is not explicitly specified by users [45, 46, 51].
- Services could behave differently when situations change, i.e. services are situation-aware. In this case, situation information is required in the matchmaking phase in service discovery for inferring the service semantics based on service descriptions.
- Situation information can be used to further categorize services for retrieving better results [50]. For example, services can be grouped by their locations, and the services close to the user are returned as the results.
- Situation information can be used in describing users' preferences to different services. For example, a user looking for hospitals may be more interested in hospitals that charge less for routine medical checkup, but will need to find the nearest hospital in the situation of medical emergency.
- Situation information can be used by service providers to control the willingness of providing services, i.e. the service providers can define policies to determine whether their services are allowed to be discovered. This is a very useful feature, especially for collaborative applications in which each user may provide some services through their mobile devices. Although many security mechanisms exist to control the access to services, a user may not even want other users to know the existence of a service running on the user's device in some situations due to performance and privacy concerns. For example, when a user's device is being used for other critical tasks, the user may want to make the services on the user's device temporarily "invisible" to other users to save resources. However, no existing discovery techniques can provide this kind of feature.

Therefore, the following issues need to be considered when designing services in mobile middleware for SA service discovery:

(1) Service semantics, users' goals and preferences, and service providers' policies under different situations should be described by appropriate languages (**R2.1** and **R2.2**).

(2) Mechanisms for matching service semantics and users' goals and preferences based on situation information need to be developed and incorporated in the matchmaking process in mobile middleware (**R2.3**).

Among existing service description languages, Web Service Description Language (WSDL) [79] and Ontology Web Language for Web Services (OWL-S) [38] are the most popular ones. WSDL is an XML-formatted language for describing a Web service's capabilities as collections of communication endpoints capable of exchanging messages. It provides a basic and simple abstraction of Web services. It is a contract or complete description that describes the components being exposed and provides names, data types, methods, and parameters required to call them. The overall structure of OWL-S includes three main parts: the service *profile* for advertising and discovering services; the *process model*, which gives a detailed description of a service's operation; and the *grounding*, which provides details on how to interoperate with a service via messages. OWL-S provides primitives for service descriptions in semantic web. However, no existing service description language provides the capability to describe situations. Some researchers have developed languages for service descriptions to support context-aware service discovery [45, 46, 80]. But their languages only provides limited capability for expressing current context, and do not have the capability to define complex situations. As discussed in Section 4.1, various languages have been developed for describe SAW requirements of mobile applications, in which necessary constructs for describing situations and applications' behavior in different situations are provided. Although these languages are not developed for describing service semantics, the way they model SAW can be adopted and combined with existing service description languages [38, 79] to support SA service discovery. Since different service providers may use different languages to describe the services they provide, or may have different interpretations on the terms that used in service descriptions even with the same language, among the existing models and languages for SAW, the ontology-based approach [11, 12, 27] appear to be good candidates to be incorporated with service descriptions.

The mechanisms for matching service semantics and users' goals and preferences based on situation information depend on how services are described. For services described using OWL, various inference engines, such as F-OWL [81] and Pellet [82], are available to provide formal reasoning support for inferring service semantics, and

matching service semantics with users' goals. In [45, 46], OWL is used for describing service ontology, and contextual information is expressed as "contextual attributes" of services. In the matching algorithm in [45, 46], concept lattices [83] are used to rate the resulting services based on their contextual attributes. In [50], a service directory is represented by a Multidimensional OEM graph [84], in which each service is a leaf node in the graph, and contextual information is used to determine which service should be used in different context. Based on such a representation of the service directory, a breadth first search algorithm is used to locate necessary services based on users' requests (goals).

- **Incorporating SAW in service coordination.**

In general, service coordination is a process of monitoring the status of participant services, invoking proper participant services, managing dependencies among participant services, and propagating necessary information to participant services to ensure the correct results obtained from the coordinated participant services. In mobile computing environments, service coordination is difficult due to the mobility of devices, which makes the set of participant services very dynamic. Recent work on context-aware service coordination [18, 66-68] has shown the advantages of coordinating services based on contextual information. Incorporating SAW in service coordination will be more beneficial since not only the current context but also the context history is considered.

As discussed in Section 2.3, existing mobile middleware provides various coordination mechanisms. Middleware based on tuple space, such as LIME [69], TSpaces [28], and Limone [70], adopts the Linda communication model [85], in which distributed processes communicate implicitly through a shared tuple space. Because of such a communication model, tuple space based middleware decouples application behavior and communications between application components, and uses the shared tuple space to store application data available to mobile units, which represents part of the context for mobile applications [86]. In [86], a case study is made based on LIME to further exploit the usage of this type of middleware to manage physical context, such as location and battery power of mobile devices, for context-aware computing. A very nice feature of this type of mobile middleware that makes them suitable for SA service coordination is that it is not necessary for applications/agents to know the participant services in advance. Applications/agents simply specify their requests and advertise the requests in the shared tuple space, and the services that can handle these requests will automatically react to the requests. Such a process seamlessly combines service discovery and coordination, and it is very easy to adapt to context/situation changes. However, a problem which needs to be addressed is that applications/agents do not have much control on the

participant services. Although applications/agents can specify the desirable characteristics of participant services, services can ignore such specifications (maliciously or carelessly) and fail the entire service coordination process.

Another coordination approach used by many other mobile middleware, such as GAIA, MobiPADS and RCSM, manages coordination based on predefined profiles or rules, and explicitly defines how contexts/situations will affect the coordination. In MobiPADS [25], service chains, which explicitly define service coordination, are defined in system profiles, associated with various contextual conditions. In runtime, service chains can be reconfigured based on current context and system profiles. In RCSM [2-4, 29, 53], service coordination is supported by inter object communications. Rules that determine when an object can interact with which object through what interface are specified during the development of applications. In these rules, situations are used as the condition for triggering the communications between objects in runtime to fulfill the application requirements. Object discovery protocols [2, 53] are provided in RCSM to address the dynamicity of mobile computing environments. This type of middleware has more control on the participant services, but it is not as flexible as tuple space based middleware due to the need of predefining profiles or rules for coordination, and it requires additional overhead for service discovery. In [13], an agent-based approach combining with AI planning techniques is introduced to enable more flexible service coordination. Figure 2 illustrates the agent-based approach for SA service coordination in [13]. It is assumed that there is a mission planner (MP) in a service-based system, which accepts mission goals specified by users, and generates execution plans based on available services and current situation. The generated execution plan is a series of service compositions to be executed in order to fulfill the overall mission goal. A step (service invocation) in the execution plan may have certain dependencies on situations, i.e., a step can be executed only when a certain situation is detected. As discussed in Section 3, it is unlikely for the MP to have complete knowledge on planning domain in real-world applications, especially in service-based mobile applications where the users are moving, central control on adding/removing services is not available, and services may be unavailable without notifying users. A consequence of planning without complete domain information is that the generated workflows may be non-executable due to dependency violations in a service

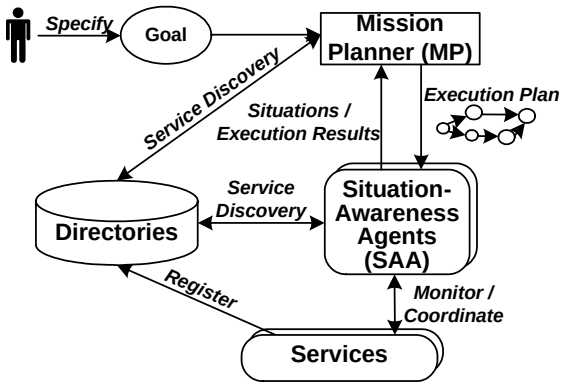


**Figure 2**. SA service coordination

invocation caused by situation changes by uncontrollable external agents. Due to this difficulty, SAW agents are developed to coordinate the execution of the services in the execution plan based on situations. SAW agents are distributed autonomous software entities, which have the necessary capabilities to support SA service coordination, including participant service management, agent discovery, and context acquisition and situation analysis.

With these capabilities, SAW agents can adaptively coordinate services in execution plans as follows:

(1) In each step of the workflow execution, SAW agents check whether all the dependencies on situations are satisfied.

(2) If the dependency on a situation is not satisfied in a certain step, the SAW agents will check whether this step can be undone.

(3) If the step is undoable, the SAW agents will first undo the step, and then search for an alternative service.

(4) If an alternative service is found, the SAW agents will resume the execution using the new service. Otherwise, the SAW agents will notify MP of and the current situation and the step having failed MP will do the re-planning and try to find another workflow that can fulfill the mission goal.

The above approaches for service coordination have their advantages and disadvantages. Further investigation is needed to evaluate and compare the complexity and performance of these approaches to determine which approaches are more effective for SA service coordination.

## 5. SUMMARY

In this chapter, we have discussed the motivation for developing mobile middleware for situation-aware service discovery and coordination for effective integration of mobile devices with service-based network infrastructures. We have presented the current state of the art on situation-awareness, service discovery and coordination, and mobile middleware. In particular, two mobile middleware RCSM and MobiPADS are discussed for providing context-/situation-awareness to application layer. The requirements, design issues and some enabling techniques for mobile middleware for situation-aware service discovery and coordination are also presented. Although context-/situation-aware mobile middleware and various techniques for context-/situation-aware service discovery and coordination have been developed, there is no mobile middleware that can fully support situation-aware service discovery and coordination. Further investigations on incorporating situation-awareness in service description languages, situation-aware matchmaking mechanisms, new coordination models, and adaptive middleware

architecture are needed for the design and development of mobile middleware supporting situation-aware service discovery and coordination. For more detailed information, the readers are referred to the references of this chapter.

## ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation under grant numbers ANI 0123980, and the DoD/ONR under the Multidisciplinary Research Program of the University Research Initiative, Contract No. N00014-04-1-0723.

## REFERENCES

[1] Web Services Architecture. Available at: *http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.*

[2] Yau, S.S. and Karim, F., A context-sensitive middleware-based approach to dynamically integrating mobile devices into computational infrastructures, *J. Parallel and Distributed Computing*, 64(2), 301, 2004.

[3] Yau, S.S., Wang, Y., and Karim, F., Development of situation-aware application software for ubiquitous computing environments, in *Proc. 26th IEEE Int'l Computer Software and Applications Conf.*, 2002, 233.

[4] Yau, S. S. et al., Reconfigurable context-sensitive middleware for pervasive computing, *IEEE Pervasive Computing*, 1(3), 33, 2002.

[5] McCarthy, J. and Hayes, P.J., Some philosophical problems from the standpoint of artificial intelligence, *Machine Intelligence 4*, 463, 1969.

[6] Pinto, J.A., Temporal reasoning in the situation calculus, PhD Thesis, University of Toronto, 1994.

[7] McCarthy, J., Situation calculus with concurrent events and narrative, 2000. Available at: http://wwwformal.stanford.edu/jmc/narrative/narrative.html

[8] Plaisted, D., A hierarchical situation calculus, *Computing Research Repository*, cs.AI/0309053, 2003.

[9] Barwise, J., Scenes and other situations, *J. Philosophy*, 77, 369, 1981.

[10] Barwise, J., The situation in logic, *CSLI Lecture Notes 17*, 1989.

[11] Matheus, C.J., Kokar, M.M., and Baclawski, K., A core ontology for situation awareness, in *Proc. 6th Int'l Conf. on Information Fusion*, 2003, 545.

[12] Matheus, C.J. et al., Constructing RuleML-based domain theories on top of OWL ontologies, in *Proc. 2nd Int'l Workshop on Rules and Rule Markup Languages for the Semantic Web*, 2003, 81.

[13] Yau, S.S. et al., Situation-awareness for adaptable service coordination in service-based systems, in *Proc. 29th Annual Int'l Computer Software and Application Conference*, 2005, to appear.

[14] Schilit, B. and Theimer, M., Disseminating active map information to mobile hosts, *IEEE Network*, 8(5), 22, 1994.

[15] Brown, P.G., Bovey, J.D., and Chen, X., Context-aware applications: from the laboratory to the marketplace, *IEEE Personal Communications*, 4(5), 58, 1997.

[16] Brézillon, P. and Pomerol, J.C., Contextual knowledge sharing and cooperation in intelligent assistant systems, *Le Travail Humain*, 62(3), 223, 1999.

[17] Dey, A.K. and Abowd, G.D., A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human-Computer Interaction*, 16(2-4), 97, 2001.

[18] Braione, P. and Picco, G.P., On calculi for context-Aware coordination, in *Proc. 6th Int'l Conf. on Coordination Models and Languages*, 2004, 38.

[19] Want, R. et al., The active badge location system, *ACM Trans. on Information Systems*, 10(1), 91, 1992.

[20] Schilit, B.N., Theimer, M., and Welch, B.B., Customizing mobile application, in *Proc. USENIX Symp. on Mobile and Location-Independent Computing*, 1993, 129.

[21] Spreitzer, M. and Theimer, M., Providing location information in a ubiquitous computing environment, in *Proc. 14th ACM Symp. on Operating System Principles*, 1993, 270.

[22] Harter, A. and Hopper, A., A distributed location system for the active office, *IEEE Network*, 8(1), 62, 1994.

[23] Nelson, B.J., Context-aware and location systems, PhD thesis, University of Cambridge, 1998. Available at: http://www.sigmobile.org/phd/1998/theses/nelson.pdf

[24] Caswell, D. and Debaty, P., Creating web representations for places, in *Proc. 2nd Int'l Symp. on Handheld and Ubiquitous Computing*, 2000, 114.

[25] Chan, A.T.S. and Chuang, S.N., MobiPADS: a reflective middleware for context-aware computing, *IEEE Trans. on Software Engineering*, 29(12), 1072, 2003.

[26] Roman, M. et al., A middleware infrastructure for active spaces, *IEEE Pervasive Computing*, 1(4), 74, 2002.

[27] Ranganathan, A. and Campbell, R.H., A middleware for context-aware agents in ubiquitous computing environments, in *Proc. ACM/IFIP/USENIX Int'l Middleware Conf.*, 2003, 143-161.

[28] Lehman, T.J. et al., Hitting the distributed computing sweet spot with TSpaces, *Computer Networks*, 35(4), 457, 2001.

[29] Yau, S.S. et al., Development and runtime support for situation-aware application software in ubiquitous computing environments, in *Proc. 28th Annual Int'l Computer Software and Application Conference*, 2004, 452.

[30] Pokraev, S. et al., Context-aware services: state-of-the-art, TI/RS/2003/137, 2003. Available at: https://doc.telin.nl/dscgi/ds.py/Get/File-27859/Context-aware_services-sota,_v3.0,_final.pdf.

[31] Mostefaoui, G.K., Pasquier-Rocha, J., and Brezillon, P., Context-aware computing: a guide for the pervasive computing community, in *Proc. IEEE/ACS Int'l Conf. on Pervasive Services*, 2004, 39.

[32] Jini architecture specification, Sun Microsystems, 2001. http://www.sun.com/software/jini/specs/

[33] Salutation architecture specification, The Salutation Consortium, 1999.
http://www.salutation.org/spec/Sa20e1a21.pdf

[34] Guttman, E. et al, Service location protocol, 1999. http://www.faqs.org/ftp/rfc/pdf/rfc2608.txt.pdf

[35] Understanding universal plug and play white paper, Microsoft Corporation, 2000.
http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc

[36] UDDI technical white paper, UDDI (Universal Description, Discovery and Integration), 2000.
http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[37] Klein, M. and Bernstein, A., Toward high-precision service retrieval, *IEEE Internet Computing*, 8(1), 2004, 30.

[38] OWL-S 1.0. http://www.daml.org/services/owl-s/1.0/

[39] W3C Resource Description Framework (RDF). http://www.w3.org/RDF/

[40] Web service modeling ontology. http://www.wsmo.org/

[41] Trastour, D., Bartolini, C., and Gonzalez-Castillo, J., A semantic web approach to service description for matchmaking of services, in *Proc. 1st Int'l Semantic Web Working Symp.*, 2001. Available at:
http://citeseer.nj.nec.com/trastour01semantic.html

[42] Paolucci, M. et al., Semantic matching of web services capabilities, in *Proc. 1st Int'l Semantic Web Conf.*, 2002, 333.

[43] Paolucci, M. et al., Using DAML-S for P2P discovery, in *Proc. Int'l Conf. on Web Services*, 2003, 203.

[44] Paolucci, M. et al., A broker for OWL-S web services, in *Proc. AAAI Spring Symp. Series on Semantic Web Services*, 2004. Available at: http://www.daml.ecs.soton.ac.uk/SSS-SWS04/40.pdf

[45] Broens, T., Context-aware, ontology based, semantic service discovery, Master thesis, University of Twente, the Netherlands, 2004.

[46] Broens, T. et al., Context-aware, ontology-based, service discovery, in *Proc. 2nd European Symp. on Ambient Intelligence*, 2004, 72.

[47] Czerwinski, S. et al., An architecture for a secure service discovery service, in *Proc. 5th Annual Int'l Conf. On Mobile Computing and Networks*, 1999, 24.

[48] Yolum, P. and Singh, M., An agent-based approach to trustworthy service location, in *Proc. 1st Int'l Workshop on Agents and Peer-to-Peer Computing*, 2002, 45.

[49] Liu, J. and Issarny, V., QoS-aware service location in mobile ad-hoc networks, in *Proc. 5th Int'l Conf. on Mobile Data Management*, 2004, 224.

[50] Doulkeridis, C., Valavanis, E., and Vazirgiannis, M., Towards a context-aware service directory, in *Proc. 4th VLDB Workshop on Technologies for E-Services*, 2003.

[51] Yau, S.S. et al., Situation-aware personalized information retrieval for mobile internet, in *Proc. 27th Annual Int'l Computer Software and Application Conf.*, 2003, 638.

[52] Debaty, P., Goddi, P., and Vorbau, A., Integrating the physical world with the web to enable context-enhanced services, *Technical Report HPL-2003-192*, HP Laboratories, 2003.
Available at: http://www.hpl.hp.com/techreports/2003/HPL-2003-192.pdf

[53] Yau, S.S. and Karim, F., An energy-efficient object discovery protocol for context-sensitive middleware for ubiquitous computing, *IEEE Trans. on Parallel and Distributed Systems*, 14(11), 1074, 2003.

[54] Stoica, I. et al., Chord: a scalable peer-to-peer lookup service for internet applications, in *Proc. ACM SIGCOMM'01*, 2001, 149.

[55] Balazinska, M., Balakrishnan, H., and Karger, D., INS/Twine: a scalable peer-to-peer architecture for intentional resource discovery, in *Proc. 1st Int'l Conf. on Pervasive Computing*, 2002, 195.

[56] Crespo, A. and Molina, H.G., Routing indices for peer-to-peer systems, in *Proc. 22nd Int'l Conf. on Distributed Computing Systems*, 2002, 23.

[57] Kashani, F.B., Chen, C., and Shahabi, C., WSPDS: web services peer-to-peer discovery service, in *Proc. Int'l Symp. on Web Services and Applications*, 2004, 733.

[58] Nidd, M., Service discovery in DEAPspace, *IEEE Personal Communications*, 8(4), 39, 2001.

[59] Chakraborty, D. et al., GSD: a novel group-based service discovery protocol for MANETs, in *Proc. 4th Int'l Workshop on Mobile and Wireless Communications Networks*, 2002, 140.

[60] Denny, M. et.al., Mobiscope: a scalable spatial discovery service for mobile network resources, in *Proc. 4th Int'l Conf. on Mobile Data Management*, 2003, 307.

[61] Helal, S. et.al., Konark – a service discovery and delivery protocol for ad-hoc networks, in *Proc. 3rd IEEE Conf. on Wireless Communications and Networking*, 2003, 2107.

[62] Berger, S. et. al., Towards pluggable discovery framework for mobile and pervasive applications, in *Proc. 5th Int'l Conf. on Mobile Data Management*, 2004, 308.

[63] Tchakarov, J. and Vaidya, N., Efficient content location in mobile ad hoc networks, in *Proc. 5th Int'l Conf. on Mobile Data Management*, 2004, 74.

[64] Web Services Coordination (WS-Coordination). Available at:
*http://www-106.ibm.com/developerworks/library/ws-coor/*

[65] Web Services Coordination Framework (WS-CF). Available at:
*http://www.oracle.com/technology/tech/webservices/htdocs/spec/WS-CF.pdf*

[66] Andrade, L.F. et al., Coordination for orchestration, in *Proc. 5th Int'l Coordination Conf.*, 2002, 5.

[67] Cabri, G., Leonardi, L., and Zambonelli, F., Engineering mobile agent applications via context-dependent coordination, *IEEE Tran. On Software Engineering*, 28(11), 1039, 2002.

[68] Julien, C. and Roman, G., Egocentric context-aware programming in ad hoc mobile environments, in *Proc. 10th Int'l. Symp. On the Foundations of Software Engineering*, 2002, 21.

[69] Murphy, A., Picco, G., and Roman, G., LIME: a middleware for physical and logical mobility, in *Proc. 21st Int'l Conf. on Distributed Computing Systems*, 2001, 524.

[70] Fok, C.L. et al., A lightweight coordination middleware for mobile computing, in *Proc. 6th Int'l Coordination Conf.*, 2004, 135.

[71] Haahr, M., Cunningham, R., and Cahill, V., Supporting CORBA applications in a mobile environment, in *Proc. 5th ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, 1999, 36.

[72] Campbell, A.T. et al., The Mobiware toolkit: programmable support for adaptive mobile networking, *IEEE Personal Comm.*, 5(4), 32, 1998.

[73] Yau, S.S., Chandrasekar, D., and Huang, D., An adaptive, lightweight and energy-efficient context discovery protocol for ubiquitous computing environments, in *Proc. 10th Int'l Workshop on Future Trends of Distributed Computing Systems*, 2004, 261.

[74] Doherty, P. and Kvarnstrom, J., TALplanner: A temporal logic-based planner, *AI Magazine*, 22(3), 95, 2001.

[75] Davulcu, H., Kifer, M., and Ramakrishnan, I.V., CTR-S: a logic for specifying contracts in semantic web services, *Proc. 13th Int'l World Wide Web Conf.*, 2004, 144.

[76] Gordon, A.D. and Cardelli, L., Equational properties of mobile ambients, *Mathematical Structures in Computer Science*, 13(3), 371, 2003.

[77] Clarke, E.M., Grumberg, O., and Peled, D.A., Model checking, MIT Press, 2000.

[78] Olston, C. and Widom, J., Best-effort cache synchronization with source co-operation, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data 2002*, 2002, 73.

[79] Web Services Description Language (WSDL) 1.1, W3C. Available at: http://www.w3.org/TR/wsdl

[80] Mostefaoui, S.K. and Hirsbrunner, B., Context aware service provisioning, in *Proc. IEEE/ACS Int'l Conf. on Pervasive Service*, 2004, 71.

[81] F-OWL home page. http://fowl.sourceforge.net/index.html

[82] Pellet OWL Reasoner home page. *http://www.mindswap.org/2003/pellet/index.shtml*

[83] Ganter, B. and Stumme, G., Formal concept analysis: methods and applications in computer science, TU Dresden, http://www.aifb.uni-karlsruhe.de/WBS/gst/FBA03.shtml.

[84] Stavrakas, Y. and Gergatsoulis, M., Multidimensional semistructured data: representing context-dependent information on the web, in *Proc. 14th Int'l Conf. on Advanced Information Systems Engineering*, 2002, 183.

[85] Gelernter, D., Generative communication in Linda, *ACM Computing Surveys*, 7(1), 80, 1985.

[86] Murphy, A. and Picco, G., Using coordination middleware for location-aware computing: a LIME case study, in *Proc. 6th Int'l Coordination Conf.*, 2004, 263.

## AUTHORS

**Stephen S. Yau** is currently a professor in the Department of Computer Science and Engineering at Arizona State University, Tempe, Arizona, USA. He served as the chair of the department from 1994 to 2001. He was previously with the University of Florida, Gainesville and Northwestern University, Evanston, Illinois. He served as the president of the IEEE Computer Society and the editor-in-chief of IEEE Computer magazine. His current research is in software engineering, distributed and service-oriented computing, adaptive middleware, and trustworthy computing. He received a BS from National Taiwan University, Taipei, and an MS and PhD from the University of Illinois, Urbana, all in electrical engineering. He is a life fellow of the IEEE and a fellow of American Association for the Advancement of Science. Contact him at yau@asu.edu or visit his website: http://www.eas.asu.edu/~seg/yau

**Dazhi Huang** is a PhD student in the Department of Computer Science and Engineering at Arizona State University. His research interests include middleware, mobile and ubiquitous computing, and service-oriented computing. He received his BS in computer science from Tsinghua University in China. Contact him at dazhi.huang@asu.edu.