Chapter 1

Service-Oriented Architecture for Mobile Services

Hong-Linh Truong and Schahram Dustdar

Contents

1.1 Introduction

Over the last few years, Service-Oriented Computing (SOC) and Service-Oriented Architecture (SOA) have demonstrated their capabilities in tackling integration and interoperability challenges raised by complex, heterogeneous data, services, and applications spanning different organizations. SOC is a computing paradigm in which services are considered as the fundamental elements for building distributed applications [1]. In SOC, SOA plays a major role to define how service-based systems are designed [1]. Various standards and technologies have been developed to support SOC/SOA concepts, such as SOAP (Simple Object Access Protocol [http://www.w3.org/TR/soap12-part1/]), WSDL (Web Services Description Language [http://www.w3.org/TR/wsdl]), and BPEL (Business Process Execution Language [http://www.oasis-open.org/committees/wsbpel/]). Among enabling technologies for SOA, Web services technologies, mostly based on XML, HTTP, WSDL, and SOAP, are the most widely implemented choice for SOA-based solutions. Web services technologies have been widely applied in small- and large-scale distributed systems built atop different platforms, such as high-end machines, mid-range servers, and workstations, for real-world applications.

As SOC/SOA offers many advantages for integrating heterogeneous data, services, and applications, SOC/SOA is a powerful enabling model for the development of mobile services. Originally, mobile services were considered as part of

telecommunication service providers but today mobile services have evolved dramatically in many aspects [2]. The term "mobile services" is typically used to indicate services for mobile users. Although mobile services are usually considered to provide Web content to mobile users from anywhere, at anytime, and with any device, as indicated by Häyrynen in *Enabling Technologies for Mobile Services: The MobiLife Book* [3], the concept of mobility is not limited to the access of Web content and services from mobile devices. In our study, mobile services offer functionality for mobile applications to *access and provide* distributed data, content, and services. Used by mobile users, mobile applications typically rely on mobile devices and mobile networks, which can be dedicated (e.g., Internet and telecommunication network) or nondedicated (e.g., a mobile *ad hoc* network—MANET) established in specific situations. Thus, it is clear that different levels of movements of users, devices and services exist, and these movements must be taken into account and supported by mobile services [4]. To date, mobile services for mobile applications hosted in dedicated infrastructures of telecommunication and Internet service providers have been widely and well developed. However, with the recent development and deployment of powerful mobile devices (e.g., personal digital assistants (PDAs) and smart phones) and mobile network infrastructures and techniques (e.g., (free) abundant WiFi hotspots and MANET), mobile services are increasingly developed and deployed on mobile devices to offer on-demand or personal services to users.

In this chapter, we study state-of-the art SOC/SOA for mobile services. We consider software services, not business services, and concentrate on mobile services based on Web services technologies[*] and on mobile devices owing to several reasons. First, various mobile applications on mobile devices have been developed based on Web services technologies because these technologies enable applications on mobile devices to access data, information, and services via standard protocols (e.g., see many StrikeIron Web services at http://www.strikeiron.com/StrikeIronServices.aspx). Although Web services technologies are largely used for developing mobile applications at the client side, supporting Web services for mobile devices on the client side is not as strong as that in other platforms. Furthermore, recently, the ways of developing and using mobile applications have been changed dramatically to meet the requirements of user participation[†] and mass customization.[‡] Therefore, it makes sense to study existing solutions to distill specific versus generic, and inflexible versus reusable, protocols and models for mobile Web services.

Second, many research efforts have focused on providing programming tools and software engineering methodologies for developing applications for mobile devices (e.g., PDAs, smart phones, subnotebooks, and laptops). In most cases, existing tools

[*] Hence, we use the term "mobile Web services" to refer to mobile services implemented with Web services technologies.
[†] http://en.wikipedia.org/wiki/Web_2.0
[‡] http://en.wikipedia.org/wiki/Mass_customization

and methodologies aim at supporting the development of mobile applications acting as a client to access data and services from dedicated, high-end systems. This is partially due to constrained resources of mobile devices on which mobile applications function, and the usage mode in which mobile devices tend to access services rather than to provide them. However, many mobile applications and scenarios have shown the need to have Web services on mobile devices [5–7]. The lack of studies of how we can reuse/integrate current mobile applications for/into existing, diverse Web services available in today's pervasive environments has also shaped our focus in this study.

Finally, mobile services that are not based on Web services or SOA-based technologies on telecommunication or Internet service providers are well developed and discussed in the literature. Some parts of mobile services systems are built on the basis of typical, dedicated distributed systems (e.g., the service side of mobile services) that are addressed well by utilizing existing SOA techniques. Therefore, we will not concentrate on nonmobile Web services and on the service side of mobile services developed in conventional distributed systems.

The rest of this chapter is organized as follows: Section 1.2 discusses why SOC/SOA for mobile services is important. Section 1.3 discusses current SOA techniques for mobile services, with a focus on fundamental architectural styles and protocols. We present mobile Web services programming supports in Section 1.4. Then, we discuss a real-world application in Section 1.5. Different SOA techniques are discussed in Section 1.6. Section 1.7 outlines research challenges. We conclude the chapter in Section 1.8. This chapter also suggests a further reading list in Section 1.9 and finally, a list of possible exercises.

1.2 Why SOC/SOA for Mobile Services?

According to the GSMA (GSM Association [http://www.gsmworld.com/]), at the time of writing the mobile world has reached 4 billion connections and by 2013 it is forecasted to have 6 billion connections [8]. With such a large number of mobile devices (and mobile users), obviously there is a strong need for accessing and sharing data, information, content, and services by using mobile devices.

Before explaining the role of SOC/SOA for mobile services, it makes sense to discuss typical mobile services. Various authors have presented different classifications of mobile services. For example, mobile services have been classified according to type of consumptions, context, social setting, and relationship between consumer and service providers in Reference [9]. As SOC/SOA is a computing model used to address the integration and interoperability challenges, it would be better to classify mobile services in a way that highlights the main benefit of SOC/SOA. Within this view, we divide mobile services into two main classes: *individual-oriented* and *group-oriented* mobile services. The first class, depicted in Figure 1.1, indicates mobile services designed and developed for individuals[*] who (occasionally) utilize the services for different purposes and have few interactions with others in close time and/or space dimensions. Here, individual-oriented mobile services are tailored for individual use. In the second class, mobile services are designed for a group of people working together in close time and/or space dimensions, shown in Figure 1.2. In this case, there are a high number of interactions among people through mobile services in a close/distributed space in a short/long period of time. This kind of mobile service might be deployed on a nondedicated infrastructure. Although this simple classification provides nonorthogonal classes, it helps us to identify some major issues that SOC/SOA can help. Table 1.1 presents some typical properties of the two classes.

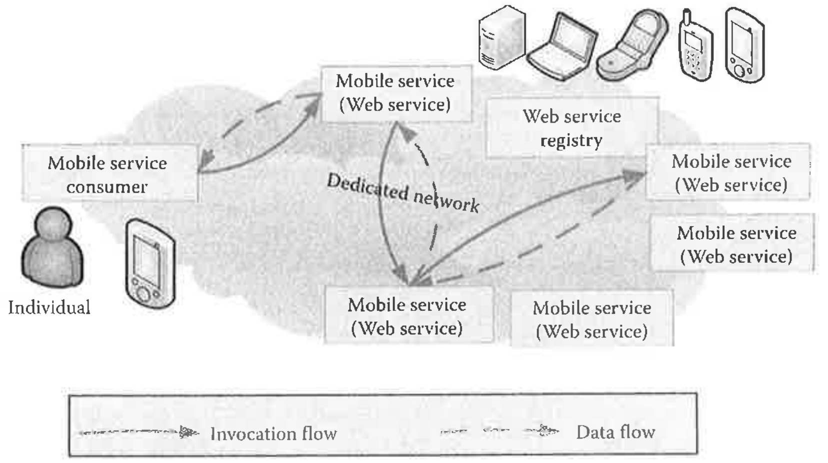

Mobile service (Web service)
Web service registry
Mobile service consumer
Dedicated network
Mobile service (Web service)
Mobile service (Web service)
Individual
Mobile service (Web service)
Mobile service (Web service)
Invocation flow
Data flow

Figure 1.1 Individual-oriented mobile services.

For the first class, we can utilize SOC/SOA to achieve the interoperability and integration of different services to provide added-value services for individuals. Let us consider diverse and rich sets of available services in the market for individual use. SOC/SOA will be useful for developing and integrating mobile applications utilizing these services due to various factors. First, there are new business models for individual-oriented mobile services which are targeted to normal people. Such models can rely on a vast source of data, contents, and services provided by different vendors through mobile networks or the Internet, to create converged services. On the one hand in today's market, data, information, and service providers want to maximize the number of their customers. Therefore, they provide their services with well-defined interfaces based on SOC/SOA models to simplify the access of

[*] We do not mean that the service is not by a single person but rather that users' usages are separated, sometimes unrelated, from each other.

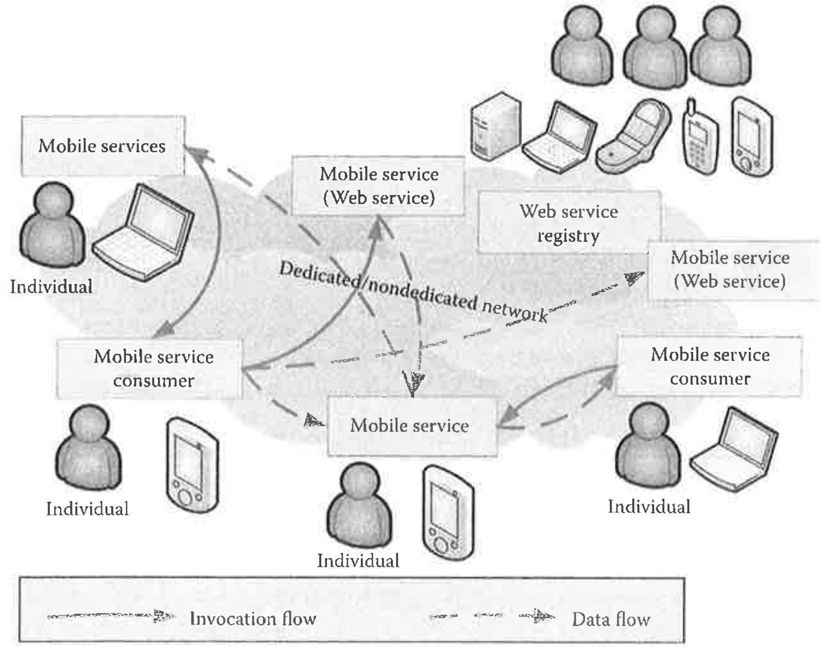


Figure 1.2 Group-oriented mobile services.

Table 1.1 Properties of Individual-Oriented and Group-Oriented Mobile Services

Properties	*Individual-Oriented*	*Group-Oriented*
Infrastructure	Dedicated infrastructure	Dedicated and nondedicated infrastructure
User involvement	Single user	Multiple users
Interaction among users	Very loosely	Tightly, concurrent, near-real-time interactions
Typical applications	E-commerce, booking, payment, travel information, e-government	Multiplayer game, collaborative work, mobile field assistant, mobile asset management, CRM (customer relationship management) field applications

their services and to foster the service composition and the integration among services to provide value-added services. On the other hand, there is a strong need to access these services by normal people during their movement, for example, booking flights, searching information, making payments, and accessing government documents. The end user also wants to participate in the Web, for example, to perform mash-ups of data and contents from different (Web) services and to provide their own mobile services. Therefore, SOC/SOA-based techniques could help mobile applications utilize SOA-based services in a standard way in order to fulfill the need of the user. Non-SOA solutions would limit mobile applications to access diverse and powerful Web services.

In the group-oriented class, there are group-oriented tasks where mobile services on mobile devices play a critical role. Examples of these tasks are collaborative work performed by a team using mobile devices [5], distributed healthcare [10,11], and disaster responses [12]. These tasks not only require access to diverse data and services hosted in distributed organizations but also need to access and share information in environments with nondedicated infrastructures. Thus, interoperable solutions for mobile services must be supported. In particular, mobile devices have been considered to be very useful in *ad hoc* team collaborations where dedicated infrastructures are not available. Such collaborations normally require flexible and interoperable applications to access as well as offer services. This raises the question of how to provide pervasive and mobile devices with middleware and applications so that the devices can provide collaboration services accessible through standard interfaces and protocols. SOC/SOA, which has introduced means to foster the interoperability, flexibility, and reusability of software, can be used to develop mobile Web services for these scenarios.

1.3 Architectural Styles and Protocols for Mobile Web Services

As well documented in the literature, in SOC/SOA, there are three fundamental, conceptual entities:

- *Service*: a service offers a concrete functionality that can be loosely coupled through the network based on a well-defined interface [13]. A service is expected to be autonomous, platform-independent and its functionality can be published and discovered [1].
- *Service consumer* (also called service client/requester): a service consumer is to consume a service. It makes requests to services and receives responses from the services.
- *Service registry*: a service registry provides facilities for services to publish information about their functionality, interfaces, and locations so that service consumers can search and select relevant services.

These roles are conceptual because an application might function as both a service and a service consumer; a service might act as a service consumer of another service; and a service registry might not be a separate element when its registry function is embedded into an application. Often, we distinguish the *client side* and *service side* when referring to service consumer components and service components, respectively.

There are various ways to design and implement SOC/SOA solutions for mobile services. As other techniques have been well discussed in different places (see a further reading list in Section 1.9), we will focus on Web services technologies. Generally, Web services and Web service consumers can be implemented using different (de facto) standards and protocols. These standards and protocols are used to define, for example, exchange message, communication protocol, service description, and security [13]. Main standards and protocols are:

- XML: is used to describe data exchanged and information about services in Web services.
- WSDL: is used to describe the interface of Web services. Through the interface description of a Web service, Web service consumers can invoke corresponding functions provided by the service.
- SOAP: defines the structure of messages exchanged between Web service consumers and Web services. SOAP is based on XML.
- HTTP: is one of the most popular communication protocols for sending and receiving messages between Web service consumers and Web services.

Although interactions between mobile Web services and their consumers are implementation specific, we can conceptually simplify these interactions into two main types, shown in Figure 1.3. In the first type, a mobile Web service consumer (depicted by the block in Figure 1.3a), through a mobile/*ad hoc* network, will access a mobile application gateway (also known as mobile Web service/application proxy in the literature) which, in turn, will process the consumer's requests and pass the requests to corresponding mobile Web services. We call this model *indirect interaction*. In the second type, a mobile Web service consumer (depicted by the block in Figure 1.3b) will utilize mobile Web services by invoking them directly. We call this case *direct interaction*. Note that direct interaction means that the requests and responses between services and their consumers may be relayed through intermediate services that perform the request/response routing but do not change the request/response content. In both cases, the mobile Web service consumer will be executed on mobile devices, whereas mobile services might or might not be deployed in mobile devices. The mobile devices hosting mobile Web services and mobile Web service consumers can be mobile phones, smart phones, PDAs, and laptops. Networks among mobile Web service consumers and mobile Web services can include the Internet, WiFi *ad hoc* networks, UMTS, EDGE, and GPRS/GMS, as well as a mix of them. Network and device capacities strongly impact on the

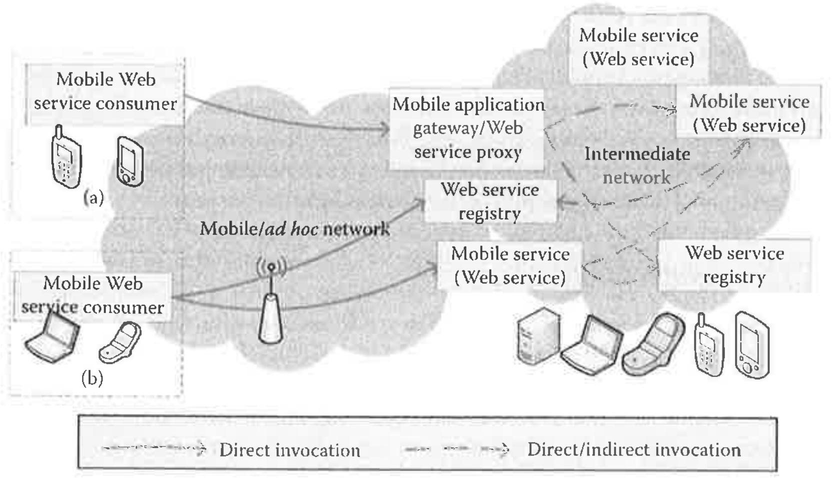

Mobile Web service consumer
(a)
Mobile Web service consumer
(b)
Mobile/*ad hoc* network
Mobile application gateway/Web service proxy
Web service registry
Mobile service (Web service)
Mobile service (Web service)
Mobile service (Web service)
Intermediate network
Web service registry
Direct invocation
Direct/indirect invocation

Figure 1.3 Generic interaction models between mobile Web service consumer and mobile services: (a) indirect interaction and (b) direct interaction.

selection of the architectural style for the design and implementation of mobile Web services.

The first type is very popular when mobile Web service consumers are executed in mobile devices that do not have enough capabilities to handle complex data types and to present results in rich, interactive graphical user interfaces. This type is also well suited for bandwidth-constrained networks. However, it does not support complex interactions among mobile users. Therefore, it is suitable for individual-oriented mobile services. In many cases, the mobile application gateway is not a Web service but is based on Web application technologies (referred to as mobile Web applications, discussed in Section 1.6.4). In the second type, the mobile Web service consumer can invoke mobile services directly as these services might provide data in a simple way or the mobile device hosting consumers have the capability to handle complex data. Furthermore, in this type, more complex interactions can be implemented, and thus it is suitable for both individual- and group-oriented mobile services.

In both types, mobile Web service consumers can invoke the service registry to find relevant services or mobile services might find relevant services on behalf of the consumer. Furthermore, services can be found from querying a dedicated registry or from an overlay network in which a dedicated registry does not exist, such as in Vimoware [5]. While mobile Web services are normally not executed on resource-constrained devices, many real-world applications have demonstrated why and how mobile services should be run in mobile devices. For example, in Vimoware [5], device sensors and context management services for disaster responses are developed as mobile Web services. Sliver [14] is another example that demonstrates how a BPEL engine can be implemented as a mobile Web service. In some cases, when a service is hosted in a mobile device, there is an intermediate to act as a proxy for other clients to access the service. For example, in the mobile service platform [15], the service consumer executes a surrogate host that acts a proxy for the service running on mobile devices. These examples indicate an increasing use of mobile Web services on mobile devices to provide service features.

1.3.1 Architectural Styles

From architectural styles, SOC/SOA Web service solutions for mobile services can be built on the basis of (i) SOAP-based Web services, (ii) REST-based Web services, and (iii) mixed SOAP/REST-based Web services with other technologies. As a specific solution can use different techniques, we will discuss only fundamental architectural styles.

1.3.1.1 SOAP-Based Web Services

In SOAP-based Web services, the interface of a Web service is described by WSDL. The messages exchanged between a Web service and its consumers are based on SOAP. SOAP-based messages are designed to be transferred by using different protocols, such as RPC, HTTP, and SMTP. However, HTTP is the most popular protocol used in SOAP-based Web services. These services achieve the integration and interoperability through agreed interfaces and, additionally, agreed service contracts. They support complex interaction models based on different styles of message-oriented and RPC communications, including point-to-point, broker, and P2P. Another advantage of this style is that interfaces can be published and searched, thus simplifying the service discovery process. Therefore, two different services provided by different organizations can easily be integrated with each other. From an architecture point of view, SOAP-based Web services are suitable for integrating mobile Web services on the service side and for group-oriented mobile services.

To date, SOAP-based Web services are strongly supported in mobile application development. Standard protocols and various programming toolkits have been developed for writing SOAP-based mobile services in mobile devices, such as PDAs and smart phones, using different programming languages, for example, C/C++, C#, and Java. However, most toolkits focus on the development of client-side components. The development and deployment of service-side components on mobile devices are not well supported, only in a few specific toolkits that offer limited features.

1.3.1.2 RESTful Web Services

REST-based Web services are built on the basis of the concept of REST (REpresentation State Transfer) [16]. REST is a design principle for Web services

that supports stateless services. Services offer their capabilities through resources, each identified by a unique URI and accessed and manipulated by using HTTP methods such as GET, POST, PUT, and DELETE. By using these four basic methods, any Web service consumer can read, update, create, and delete resources offered by a REST-based Web services. REST techniques offer more simple mechanisms for developing Web services by not relying on a large set of standards and by aiming at supporting request–reply interactions. Therefore, the REST architectural style fits very well to many mobile applications, in particular to individual-oriented ones that are typically request–reply-based client/service and stateless. Furthermore, as REST-based Web services mainly utilize HTTP, they are quite suitable for mobile devices because they use less processing power. This is proved through a wide range of real mobile applications based on REST such as commerce [17], data mash-up [18], and GIS for enterprise field workers [19]. However, in many cases, if we need to support more complex interaction models, such as P2P interactions, then REST is not suitable. Further discussion on the choice of REST solutions over SOAP solutions can be found in Reference [20].

1.3.1.3 Mixed Architectural Styles

In many cases, mobile services solutions are not completely based on either SOAP or REST Web services but a mix of them, and they also use other technologies. Figure 1.4 shows some possible mixed architectural styles used for mobile Web services.

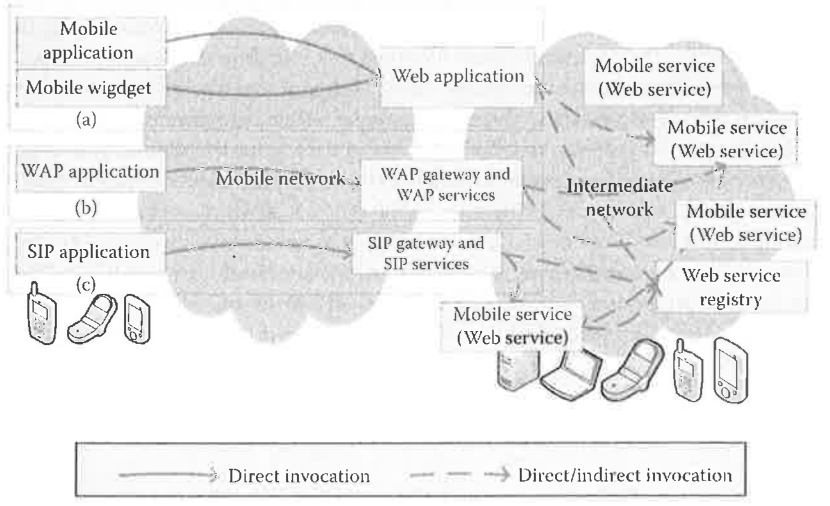

Mobile application
Mobile wigdget
(a)
WAP application
(b)
SIP application
(c)
Web application
Mobile network
WAP gateway and WAP services
SIP gateway and SIP services
Intermediate network
Mobile service (Web service)
Mobile service (Web service)
Mobile service (Web service)
Web service registry
Mobile service (Web service)
Direct invocation
Direct/indirect invocation

Figure 1.4 Mixed architectural styles for mobile service-oriented architecture services.

- Using a REST-based model for connecting a client to a gateway and using SOAP-based services for service integration at the service side: First, this architecture aims at addressing the simplicity and performance in the interaction between mobile service consumers and mobile services. Second, by using SOAP-based styles at the back end of the service side, the service provider can integrate different types of services from different providers to offer many benefits to consumers.
- Using SIP (Session Initiation Protocol), Web applications, and WAP (Wireless Application Protocol) between consumers and appropriate gateways, and SOAP-based solutions for integrating gateways with other services: This approach is able to deal with limited capability and infrastructure available at the client site to provide converged mobile services [7], such as voice communication integrated with Web content and IP multimedia subsystem application servers [21].

1.3.2 Communication Patterns

Communication models for mobile services naturally follow existing models such as request–reply, broadcast/multicast, and P2P. The design of SOAP allows different one-way or two-way, asynchronous/synchronous, one-to-one, one-to-many, broadcast, P2P communication patterns [22]. The most common communication patterns implemented in SOAP-based mobile Web services are synchronous request–reply, multicast, and P2P using RPC, message-oriented communication, and intermediate services. However, similar patterns have not been observed in REST-based Web services that mostly support only request–reply patterns.

Figure 1.5 illustrates some patterns on service invocation. In the request–reply pattern, a mobile service consumer sends a request to a mobile Web service and obtains the response. This communication can be synchronous or asynchronous, but a synchronous request–reply pattern is the most popular one implemented in contemporary mobile Web services applications, both in SOAP and REST styles, such as in References [23–26]. This pattern is suitable for individual-oriented mobile services; however, it does not work well in group-oriented mobile services of which mobile applications need to be informed with new information instantly, such as in a situational change in disaster responses. Asynchronous request–reply, broadcast/multicast, and P2P patterns are typically supported through specific SOAP/REST-based call-back and polling techniques, asynchronous HTTP, or WS-Notification implementations, such as in References [5,27]. They can be implemented well with SOAP-based mobile services. However, most programming toolkits require the developer to do this without any support. The one-way invocation pattern is also supported by SOAP standards.

1.3.3 Service Discovery and Composition

Service discovery and composition are fundamental processes of SOA. To support these processes, services have to be well defined and described. In general, with

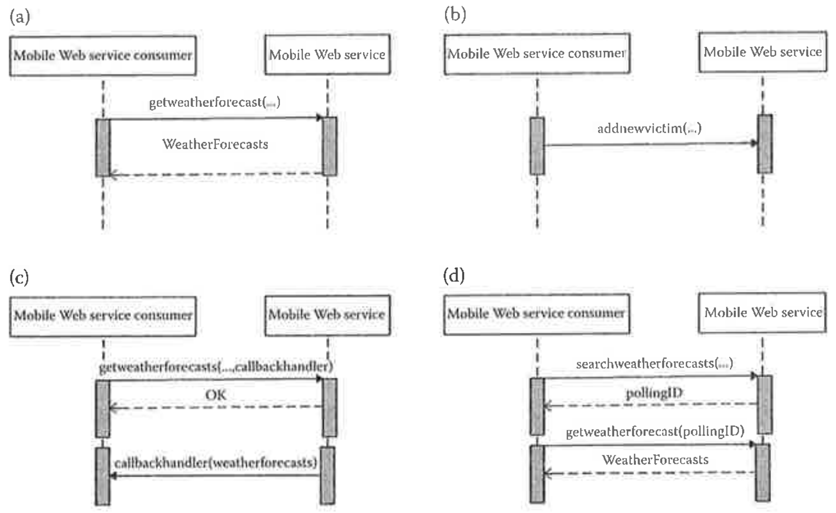

(a)
Mobile Web service consumer
Mobile Web service
getweatherforecast(...)
WeatherForecasts
(b)
Mobile Web service consumer
Mobile Web service
addnewvictim(...)
(c)
Mobile Web service consumer
Mobile Web service
getweatherforecasts(...,callbackhandler)
OK
callbackhandler(weatherforecasts)
(d)
Mobile Web service consumer
Mobile Web service
searchweatherforecasts(...)
pollingID
getweatherforecast(pollingID)
WeatherForecasts

Figure 1.5 Examples of invocation patterns. (a) Synchronous request-replay invocation. (b) One-way invocation. (c) Asynchronous invocation using callbacks. (d) Asynchronous invocation using polling.

SOAP-based services, the use of WSDL and other metadata has facilitated the discovery process. Semantic representations of mobile Web services are also used increasingly. In principle, mobile Web service consumers find SOAP-based Web services from service registries, such as Universal Description, Discovery, and Integration (UDDI). In practice, this model does not work well [28]. However, the REST-based Web services discovery process is largely negligible at the moment. In fact, there is a lack of mechanisms to describe REST-based Web services that can facilitate the discovery process. One of few efforts is the proposal of Web Application Description Language (WADL) [29] for describing REST resources. However, it has not been widely adopted.

Figure 1.6 shows basic models of Web services discovery that work with mobile Web services. For multicast discovery, these models can be supported by SLP (Service Location Protocol [http://tools.ietf.org/html/rfc2608]), UPnP (http://www.upnp.org), WS-Discovery (http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf), and tool-specific implementation, and can be implemented using UDP/HTTP multicast, SOAP, and WS-Notification. As mobile devices have some limitations, in many cases, mobile Web service consumers have not actually implemented the discovery process, especially in individual-oriented mobile services. The user just provides his/her known services (based on other sources, e.g., search engines) to mobile applications, for example, as in Reference [30]. For group-oriented mobile services, various attempts

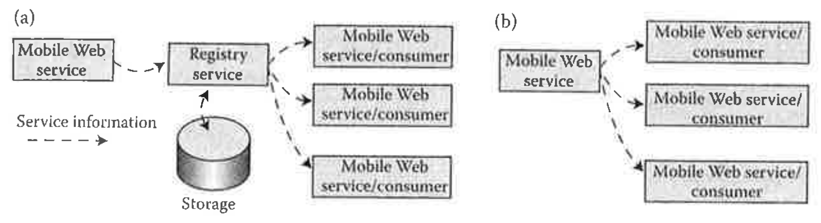

(a)
Mobile Web service
Registry service
Mobile Web service/consumer
Mobile Web service/consumer
Mobile Web service/consumer
Service information
Storage
(b)
Mobile Web service
Mobile Web service/consumer
Mobile Web service/consumer
Mobile Web service/consumer

Figure 1.6 Examples of discovery models. (a) Centralized and multicast discovery with dedicated registry. (b) Multicast discovery without dedicated registry.

have been made to support broadcast/multicast and P2P service discovery. A JXTA-based P2P discovery for mobile Web services is presented in References [31,32]. In another effort, SLP-based/specific discovery techniques have been developed in RESCUE [33,34]. The WSAMI middleware [24] also supports an SLP-based discovery. Broadcast/multicast and SLP-based discovery methods are useful for mobile services in a network; however, they do not scale well in a large-scale system of mobile services. Therefore, they are more suitable for a small group of services.

The composition of mobile Web services occurs mostly on the service side in nonmobile platforms by using well-known techniques. However, service aggregation techniques on mobile devices are very limited. Some efforts have been devoted to service aggregation by utilizing workflows [35], in particular using BPEL, in the mobile devices, for example, Sliver [14]. This type of service aggregation is an attempt to support the user to coordinate tasks and to perform business processes using mobile devices. To support the user to compose his/her own services, some research efforts have been investigated [36]. However, currently research issues in this direction are very open.

1.3.4 Data Handling

Data handling in mobile applications is strongly dependent on specific services and device capabilities. Typical data handled by mobile applications are Web contents and multimedia. Various standards have been proposed for handling these types of data, such as Mobile Media application programming interface (API) (JSR 135), Mobile 3D Graphics API (JSR 184), and 2D Scalable Vector Graphics API (JSR 226). Many types of mobile Web services do not transfer rich data or large amounts of data. However, recent advanced mobile Web services, such as location-based services and GIS-based collaborative tools, require the transfer of rich data. In particular, many mobile Web services support user participation and mass customization by offering a mash-up of contents from different sources; these mash-ups possibly include text, graphics, audio, and videos.

As handling complex, voluminous data transferred through Web services require strong processing capabilities, mobile Web services have to utilize different techniques to improve the data handling. One technique is not to use XML but a different data format exchanged between mobile Web service consumers and mobile Web services. A popular technique that works well with REST services is to utilize JSON (JavaScript Object Notation [http://www.json.org/]) which requires less processing capability and has smaller data size. Listing 1.1 presents an example of JSON-based data returned by the REST-based ImageSearch Service from Yahoo! (http://search.yahooapis.com/ImageSearchService/V1/imageSearch) and Listing 1.2 presents a sample of code to process the JSON-based data on mobile devices using the JSONObject library (http://www.json.org/java/index.html). Another technique is to compress the SOAP body transferred using compression techniques [37]. This technique, however, requires the control of both mobile Web service consumers and mobile Web services. Therefore, it is not interoperable. Another way is to utilize

```
{"ResultSet": {
  "Result": [
          {
               "ClickUrl": "http://www.residenzjoop.com/files/
               image/lage/hundertwasser.jpg",
               "FileFormat": "jpeg",
               "FileSize": 104448,
               "Height": "429",
               "RefererUrl": "http://www.residenzjoop.com/
               location",
               "Summary": "Hundertwasserhaus",
               "Thumbnail": {
                    "Height": "97",
                    "Url": "http://thm-a01.yimg.com/
                    image/0acfb0befb605e0e",
                    "Width": "145"
               },
               "Title": "hundertwasser jpg",
               "Url": "http://www.residenzjoop.com/files/
               image/lage/hundertwasser.jpg",
               "Width": "640"
          },
     //....
     ],
     "firstResultPosition": 1,
     "totalResultsAvailable": "556",
     "totalResultsReturned": 10
}}
```

Listing 1.1 Simplified example of JSON-based data.

```
//...
ImageSearch imagesearch = new ImageSearch("YahooDemo",
"Hunderwasserhaus", "json");
String result = imagesearch.submit();

JSONObject resultJSON = new JSONObject(result);
JSONObject resultSetObject = resultJSON.
getJSONObject("ResultSet");
int totalResultsReturned = resultSetObject.
getInt("totalResultsReturned");
if (totalResultsReturned < 1) {
       return ;
}
JSONArray resultObject = resultSetObject.
getJSONArray("Result");
for (int i = 0; i < resultObject.length(); i++) {
  JSONObject objectItem = resultObject.getJSONObject(i);
  Iterator iterator = objectItem.keys();
  while (iterator.hasNext()) {
      String key = (String) iterator.next();
      String data = objectItem.getString(key);
      if (!key.equals("Thumbnail")) {
          System.out.println(key + ":\t" + data);
      }
  }
}
//...
```

Listing 1.2 Example of processing JSON-based data.

binary formats of XML (http://en.wikipedia.org/wiki/Binary_XML) to reduce the size of transferred data.

1.3.5 Service Management

The management of mobile Web services is naturally similar to that of general Web services such as accessing service information, monitoring service status, and supporting runtime deployment. Various specifications have been developed for managing Web services, in particular SOAP-based Web services, such as WS-Management (http://www.dmtf.org/standards/wbem/wsman) and Web Services Distributed Management (WSDM) (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm#overview). These specifications focus on the access and exchange of management information of services. However, to date these standards have not been well supported in existing mobile Web services.

Reference [23] is one of the few works that provide service information using rules and notifications.

Different from other platforms, service management for mobile Web services has to deal with two major issues: service continuity and service runtime deployment and execution. Service continuity is of paramount importance in mobile services because the high level of mobility increases the high level of service disruption and mobility. Service runtime deployment and execution are other important issues because mobile devices are typically not assumed to support preinstalled services or always-on, preprovisioning services. This is particularly true when services are executed on a nondedicated infrastructure and mobile devices.

The importance of service continuity for mobile services has been identified and a modeling technique has been proposed in Reference [38]: a service continuity layer is proposed to handle tasks to ensure the continuity of services such as monitoring and handover management. In the Suspend–Relocate–Resume for Web Services (SRR-WS) framework [39], service continuity is ensured by suspending services and relocating client's data and resuming services with the client data. SRR-WS achieved this mechanism by utilizing the proxy model. To work with SRR-WS, mobile Web service consumers need to include a specific SRR-WS client library that handles caching and session relocation. The SRR-WS framework includes a proxy with suspend/resume and session relocation modules. By using a proxy mechanism, SRR-WS does not require a change in mobile services. However, consumers have to be reprogrammed to work with SRR-WS. SRR-WS has been implemented with SOAP over HTTP for mobile devices. Another technique is to migrate mobile Web services when the hosting environment is unable to host the services, for example, due to low battery or the mobility of the service provider [26]. In this way, migration requests are explicitly made by the service provider (which is also hosted on a mobile device). In principle, service code and description can be migrated. However, this solution cannot work alone without ensuring correct information about services.

Service runtime deployment and execution for mobile applications have been researched intensively. As mobile Web service consumers are part of mobile applications, it is expected that existing runtime deployment and execution techniques could be applied to mobile Web services.

1.3.6 Security and Identity Support

Security and identity support are of paramount importance for mobile users and are strongly dependent on the architectural styles used for mobile Web services. With SOAP-based solutions, various security standards have been proposed, such as the OASIS WS-Security and the Liberty ID-WSF/ID-SIS. The REST-based solutions typically rely on HTTP and SSL security techniques. Furthermore, depending on the integration of mobile Web services, different identity management techniques can be used.

For SOAP-based Web services, the OASIS WS-Security (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss) specifies protocols for securing message exchanges in Web services. WS-Security utilizes XML Digital Signature, XML Encryption, and X.509 certificates. Its key feature is to protect SOAP messages exchanged. The support of WS-Security standards on mobile Web service development toolkits is strong. Most toolkits support XML signatures and encryption. For REST-based Web services, HTTP Basic/Digest and SSL are popular techniques for implementing authentication, authorization, and encryption in mobile Web services because the communication of REST-based Web services is based on HTTP. Because these techniques are well supported in mobile device operating systems and programming libraries, they can be easily used by REST-based mobile Web services and clients. Furthermore, one practical method applied to the authentication of the access to REST-based Web services is to use tokens embedded in service interfaces (e.g., user ID, email, and development key), such as in StrikeIron and Amazon Web services.

The OAuth (http://oauth.net) is also increasingly used for authorizing resource access in mobile Web services. The OAuth is an authorization delegation protocol that defines how a user can grant service consumers access to a user's private resources hosted in another service. The OAuth is built atop HTTP and works on the basis of the exchange of two types of tokens asking for authorization access and for accessing resources on behalf of users. The protocol includes three main steps (see details in http://oauth.net/core/1.0/) to allow the consumer to obtain an unauthorized request token that is authorized by the user, to use the request token to obtain an access token, and finally to access resources. All requests are signed and parameters of requests can be described in the HTTP Authorization header, HTTP POST body, or URLs of HTTP GET. At the time of writing, the OAuth has been implemented in many programming languages, such as C#, Java, JavaScript, Python, and Ruby. As OAuth is designed for HTTP and Web resource access, it is very suitable for REST-based Web services.

Identity management techniques are critical when a mobile Web service consumer utilizes different mobile Web services, potentially provided by different providers. Being able to associate user identity with his/her selected mobile service, these services can improve the authentication and authorization, reduce user intervention, such as support single sign-on, and improve service provisioning. Two popular identity management frameworks are the ID-WSF/ID-SIS and the OpenID. The IdentifyWeb Services Framework (ID-WSF) and the Identity Service Interface Specifications (ID-SIS) (http://www.projectliberty.org/liberty/resource_center/specifications) define mechanisms enabling identity-based services. By associating the identity of individual users with Web services, ID-WSF/ID-SIS-enabled systems can establish a federated identity. By doing so, the user does not need to provide his/her identity many times when using different services and the service provider can optimize the composition of services. The ID-WSF version 2.0 supports SAML, authentication, single sign-on, and identity mapping by utilizing XML-Signature, XML-Encryption, and WS-Security.

The OpenID (http://openid.net) is another protocol to support single sign-on. OpenID is strongly supported by the industry (see the list of providers at http://openiddirectory.com/openid-providers-c-1.html). The idea of OpenID is to provide a single identity management protocol for accessing resources from different service providers. In the OpenID version 2.0 (http://openid.net/specs/openid-authentication-2_0.html) a user is identified by an OpenID identifier described in an URL or XRI. An OpenID identifier is managed by an identity provider or OpenID provider that carries out the user authentication. When a user accesses a service that offers OpenID authentication, the service will request the user to provide the user OpenID. The service will communicate with the identity provider of the user to verify the user. In case an authentication is required, the service will redirect the user to the identity provider for authentication (e.g., entering user name and password). OpenID utilizes HTTP, SSL/TSL, and URL; thus it is quite suitable for REST-based services. As OpenID focuses on authentication and single sign-on, it is suitable for service integration on the service side.

Although ID-WSF/ID-SIS and OpenID are widely supported on the service side of mobile Web services in many platforms, their availability on mobile devices where applications act on behalf of the user is limited. OpenID has been supported in many libraries but it has not been well integrated with SOAP/REST. For example, it is relatively easy to send an OpenID request to a Web service; however, authentication requiring user intervention is not quite clear (see a discussion on OpenID binding to SOAP/REST at http://wiki2008.openid.net/REST/SOAP/HTTP_Bindings).

1.4 Mobile Web Services Programming Support

1.4.1 Standards and Specifications

Various standards and specifications have been developed for Web services. However, not all of them are fully supported in mobile Web services. Fundamental standards and specifications such as SOAP, XML, and HTTP are well supported in most mobile platforms. Other standards such as OASIS WS-Security and Liberty ID-WSF are supported in some platforms.

Apart from the above-mentioned generic standards and specifications for Web services, some have been specifically designed for mobile Web services. The J2ME Web services specification (JSR-172) (http://jcp.org/en/jsr/detail?id=172) defines optional packages for the development of Web service applications on mobile devices. In particular, it focuses on XML processing capabilities and on APIs for J2MS Web service clients based on RPC communication. JSR-172 also defines the mapping from a subset of the WSDL to Java code, suitable for J2ME. This specification has been supported in many programming toolkits such as Nokia Web services platform, Sun J2ME Wireless Toolkit (http://java.sun.com/products/sjwtoolkit/), and Netbeans (http://www.netbeans.org).

The mobile service architecture (MSA) specification (JSR 248) has been introduced recently to create an environment for Java-enabled mobile devices. MSA includes several existing standards for handling security and commerce, graphics, communications, personal information, and application connectivity.

1.4.2 Mobile Web Service Toolkits

1.4.2.1 General-Purpose Web Service Toolkits

These toolkits aim at addressing broad mobile Web services applications. The main toolkits that have been developed are discussed in this section.

1.4.2.1.1 kSOAP2

kSOAP2 [40] is one of the very first and popular SOAP Web service libraries for developing Web services on constrained devices. kSOAP provides various facilities that have been utilized by other SOC/SOA work on mobile devices, such as Sliver [14], a BPEL engine, and RESCUE [33,34]. It can be used to develop both Web service clients and services. However, kSOAP does not provide advanced features for handling security, service discovery, and service management. In principle, all of these features have to be implemented by the developer.

1.4.2.1.2 gSOAP

gSOAP is an open-source C/C++ Web service development toolkit [41]. gSOAP supports XML data-binding solutions through autocoding techniques. It also provides tools for generating C/C++ code from WSDL/XSD files and supports different platforms including mobile devices. gSOAP is well studied in Reference [42]. Although gSOAP provides security support, similar to kSOAP it focuses mainly on the development of Web services and their consumers. Therefore, it does not support service discovery and aggregation.

1.4.2.1.3 Google Android

Android (http://code.google.com/android) is not a particular mobile Web service toolkit but a software platform including an operating system, middleware, and applications for mobile devices. Android provides various facilities for developing mobile applications but it does not provide a toolkit for writing SOAP-based Web services and clients. However, it provides various classes for writing mobile Web clients by providing HTTP, XML, JSON, and OAuth classes for implementing mobile Web service clients. The clients can use HTTP actions to invoke REST-based Web services. Both synchronous and asynchronous HTTP invocations are supported. The returning results can be in XML, JSON, RSS, Atom, and so on.

1.4.2.1.4 .NET

The Microsoft .NET Compact Framework (CF) (http://msdn.microsoft.com/en-us/netframework/aa497273.aspx) supports the development of applications on smart/mobile devices. The .NET CF provides various facilities for developing mobile applications, including mobile Web service applications on the client side based on HTTP, SOAP, and XML. It supports both asynchronous and synchronous invocations of Web services, HTTP Basic/Digest authentication, SSL, and SOAP extension. However, the development of Web services on mobile devices is not supported. The .NET CF also provides useful tools to support the developer to write mobile Web service clients. For example, the .NET CF Service Model Metadata Tool (netcfsvcutil.exe) can be used to generate client proxy codes to simplify the development of clients consuming Web services on the device.

1.4.2.1.5 Java FX Mobile

The Sun Java FX (http://www.javafx.com) is a platform for developing rich Internet applications and Java FX Mobile is the version of Java FX for mobile devices. The key technologies provided by Java FX are in a rich set of libraries for handling graphics, media data, and Web services. By utilizing Web service libraries, we can develop mobile service applications on mobile devices. Java FX Mobile provides only HTTP-based libraries for the development of Web service applications on the client side. Therefore, only REST-based Web service clients can be built. This library includes APIs for handling HTTP requests and XML/JSON. Java FX Mobile does not include facilities for writing REST-based Web services. Java FX is well integrated into different programming development environments such as NetBeans and Eclipse IDE.

1.4.2.1.6 Nokia Web Service Platform

The Nokia Web Services Platform [43] includes various development facilities for mobile service applications on Nokia's S60 or Series 80 platform. It supports both Java technology and Symbian OS C++. Apart from other facilities, for Java-based applications, the Nokia WSP supports the J2MEWeb Services Specification (JSR-172). At the moment, only client-side functionality is supported. The C+ version provides APIs for handling requests to and responses from Web services. These APIs are built on the basis of a Web Services Framework (WSF) API or the Liberty

Identity-Based Web Services Framework (ID-WSF). OASIS WS-Security and Liberty ID-WSF are supported.

1.4.2.1.7 Apache Muse

The Apache Muse Project (http://ws.apache.org/muse/) implements the Web Services Resource Framework (WSRF), Web Services BaseNotification (WSN), and Web Services Distributed Management (WSDM) specifications. Therefore, it supports SOAP-based Web services based on WSRF. By using WSN, Web services based on Muse can also implement eventing based on Web services. The WSDM allows the developer to include management features in Web services, for example, service capabilities and status. Muse can be deployed in different platforms including Java SE/Java EE and Open Services Gateway initiative (OSGi). When deployed in OSGi, Muse can be used to develop mobile Web services.

Table 1.2 summarizes some properties of the above-mentioned general-purpose mobile Web service toolkits.

1.4.2.2 Specific Toolkits

1.4.2.2.1 Vimoware/RESCUE

Vimoware [5] is a Web service-based toolkit that can be used to develop Web services on mobile devices and to conduct *ad hoc* team collaborations by executing predefined or situational flows of tasks. One of the main components is a *light-weight Web services middleware* that supports the SOAP-based Web services. This middleware was later developed into a separate middleware named RESCUE [33,34]. In Vimoware/RESCUE, Web services are developed by applying the POJO (Plain Old Java Object) principle. The developer creates a Web service by extending an abstract Java class. This class requires the specification of a service description and provides basic methods for extracting metadata about the service. Service operations are implemented as normal Java methods. On the basis of the description provided by the service developer and on the metadata, WSDL files can also be created.

The main specific characteristic of Vimoware/RESCUE is that it is specifically designed for Web services in the *ad hoc* network of mobile devices. Thus, it also provides runtime and reconfigurable service provisioning and service discovery facilities. Vimoware/RESCUE uses kSOAP2 and reuses parts of Sliver [14] to deploy Web services. The transport communication can be configured either by HTTP, which is realized by a light-weight version of the Jetty (http://jetty.mortbay.org) engine, or by direct TCP socket communication. In Vimoware/RESCUE, three interaction patterns exist: (a) one-way interactions, (b) synchronous request–response interactions, and (c) real asynchronous request–response. Services can be deployed into Vimoware/RESCUE at runtime. In Vimoware/RESCUE, a

Table 1.2 Summary of General-Purpose Mobile Web Service Toolkits

Toolkit	*Architectural Style*		*Client/Service*		*Language*			*Security/Identity*			*Management*
	SOAP	*REST*	*Client*	*Service*	*C/C++*	*Java*	*C#*	*WS-Based*	*HTTP-Based*	*Identity*	
kSOAP/kXML	×		×	×		×					
gSOAP	×		×	×	×			×			
Android		×	×			×			×	×	
.NET	×	×	×		×		×	×	×		
Java FX mobile		×	×			×			×		
Nokia WSP	×		×	×	×	×		×	×	×	
Muse	×		×	×		×		×			×

P2P-based subscription/notification mechanism is supported for service advertisement and discovery. This mechanism is implemented on the basis of UDP multicasts.

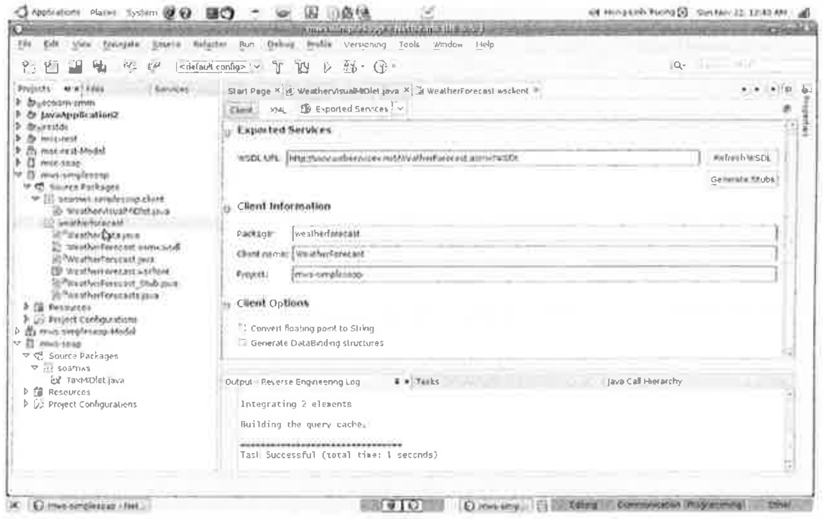
1.4.3 Software Development Process for Mobile Web Services

Unlike Web services on other platforms, the design and development of mobile Web services lack supporting tools. Model-driven development (MDD) techniques that seem very suitable for modeling diverse mobile platforms have been investigated for mobile applications [44,45]. However, unlike the support of MDD for other types of applications, MDD for SOA-based mobile Web services has not been well studied. SPATEL [46], developed in the SPICE project, is a UML-based language for modeling composite telecommunication services. This language has been demonstrated with different GUI frameworks like S60 Nokia smart phones. In Reference [45], from a meta-model, DSL (domain-specific language) is generated and can be used for domain-specific modeling (DSM). In DSM, mobile services are modeled and XForm code can be generated for use in mobile applications. However, this MDA approach is implemented for a specific platform that is not Web services.

In Reference [38], a composition model for mobile services (not necessary mobile Web services) has been proposed to include four main components, namely, service logic (describing service functions), service data (describing data used in services), service content (describing data product of services), and service profile (describing user/device properties). From the composition model, components of a mobile service can be mapped into different distribution models, representing the concrete deployment of components of mobile services, such as monolithic, client–server, peer-to-peer, or multiple distributed components [38]. This conceptual model can be applied for modeling mobile Web services. However, we are not aware of any existing tool to support this model.

1.4.4 Writing Mobile Web Services and Consumers

The writing code for mobile Web service consumers to invoke mobile Web services is straightforward and similar to that for Web services in other environments. Most programming toolkits support very primitive code generation for SOAP-based service consumers. A typical way of writing a SOAP-based service consumer code is to import the WSDL file of the service and generate a stub code from that WSDL. Figure 1.7 shows a simple example of how Netbeans supports the writing of a mobile Web service consumer code for the `WeatherForecast` service (http://www.webservicex.net/WeatherForecast.asmx?wsdl). By indicating the URL of the conversion service, a code can be generated, for example, as shown in Figure 1.8.



Exported Services

WSDL URL: [illegible]

Client Information

Package: weatherforecast

Client name: WeatherForecast

Project: [illegible]

Client Options

☐ Convert floating point to String

☐ Generate DataBinding structures

```
Integrating 2 elements
Building the query cache.
================================
Task Successful (total time: 1 seconds)
```

Figure 1.7 Example of generating code for a mobile Web service consumer in Netbeans.

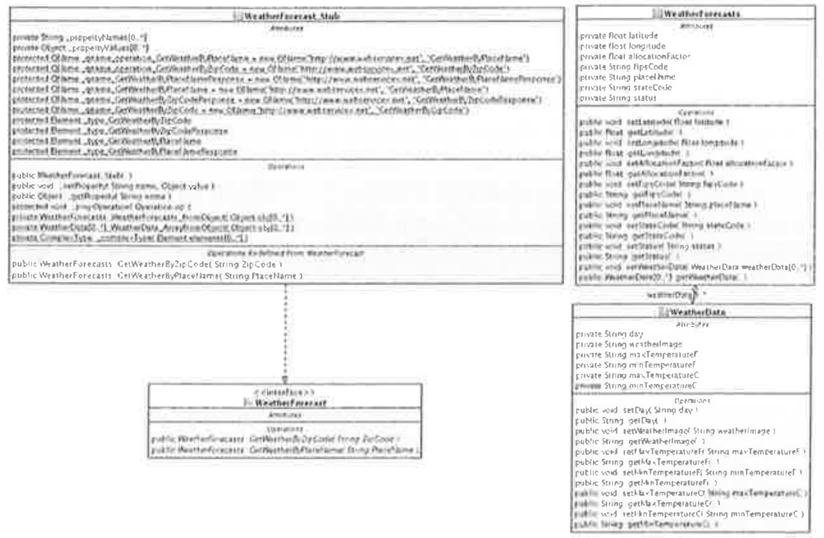

WeatherForecast_Stub
public WeatherForecasts GetWeatherByZipCode(String ZipCode)
public WeatherForecasts GetWeatherByPlaceName(String PlaceName)
WeatherForecasts
private float latitude
private float longitude
private float allocationFactor
private String fipsCode
private String placeName
private String stateCode
private String status
weatherData
WeatherData
private String day
private String weatherImage
private String maxTemperatureF
private String minTemperatureF
private String maxTemperatureC
private String minTemperatureC
public void setDay(String day)
public String getDay( )
public void setWeatherImage(String weatherImage)
public String getWeatherImage( )
public void setMaxTemperatureF(String maxTemperatureF)
public String getMaxTemperatureF( )
public void setMinTemperatureF(String minTemperatureF)
public String getMinTemperatureF( )
public void setMaxTemperatureC(String maxTemperatureC)
public String getMaxTemperatureC( )
public void setMinTemperatureC(String minTemperatureC)
public String getMinTemperatureC( )
<<interface>>
WeatherForecast

Figure 1.8 Example of the class diagram for generated code produced by Netbeans.

```
 //...
String inputZIP;
//...
try {
   WeatherForecast serviceConnection = new WeatherForecast_
Stub();
   WeatherForecasts forecasts = serviceConnection.
GetWeatherByZipCode(inputZIP);
   System.out.println(forecasts.getFipsCode());
   System.out.println(forecasts.getLatitude());
   System.out.println(forecasts.getLongitude());
   System.out.println(forecasts.getPlaceName());
   System.out.println(forecasts.getStateCode());
   System.out.println(forecasts.getStatus());
   WeatherData[] data = forecasts.getWeatherData();
   for (int i = 0; i < data.length; i++) {
          System.out.println(data[i].getDay() + ": max =" +
data[i].getMaxTemperatureC() + "min=" + data[i].
getMinTemperatureC() + "," + data[i].getWeatherImage());
   }
} catch (RemoteException ex) {
     ex.printStackTrace();
}
```

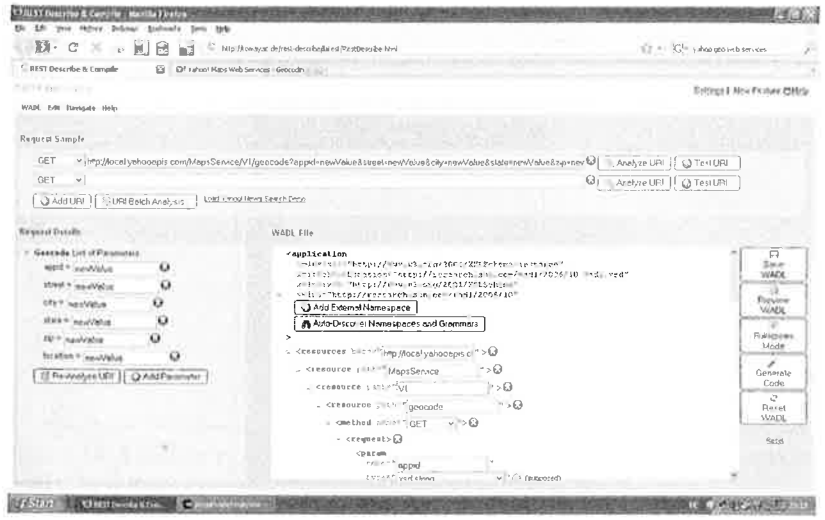
Listing 1.3 Code excerpt for invoking the `WeatherForecast` service.

A similar process can be found in other toolkits like Microsoft Visual Studio (for .NET/C#) or Eclipse.

Given the generated code, it is straightforward to write a service invocation code based on synchronous request–reply patterns; for example, Listing 1.3 shows how to invoke the `WeatherForecast` service.

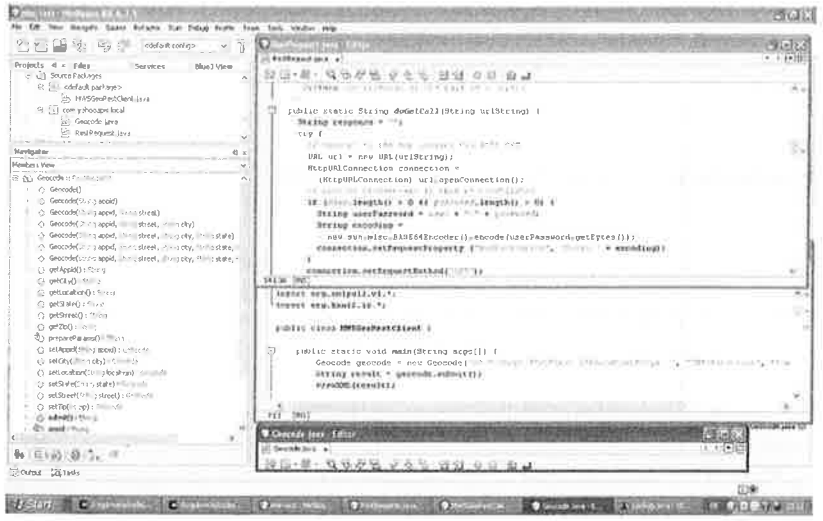
A similar software development process for REST-based mobile Web service consumer code generation is not well supported in current tools. In most cases, the developer just writes the code to invoke the service. However, recently some tools also support writing WADL-based information for REST services and code generation from WADL files. These tools should be incorporated into/combined with existing IDE for code development. For example, Figure 1.9 shows the REST Describe tool (http://tomayac.de/rest-describe/latest/RestDescribe.html) that is used to generate WADL for the `Yahoo! Geocoding Service`. On the basis of the generated WADL, this tool can generate codes for invoking the service, as shown in Figure 1.10. Figures 1.11 and 1.12 illustrate the resulting service invocations for the above-mentioned examples in different platforms.

To develop a Web service on mobile devices is much more challenging because of the lack of tools. Recently, the Apache Muse supports the developer to develop Web services in mobile devices in a way similar to normal environments. With the



```
<application
...
<resources base="http://local.yahooapis.c">
  <resource path="MapsService">
    <resource path="V1">
      <resource path="geocode">
        <method name="GET">
          <request>
            <param
              name="appid"
```

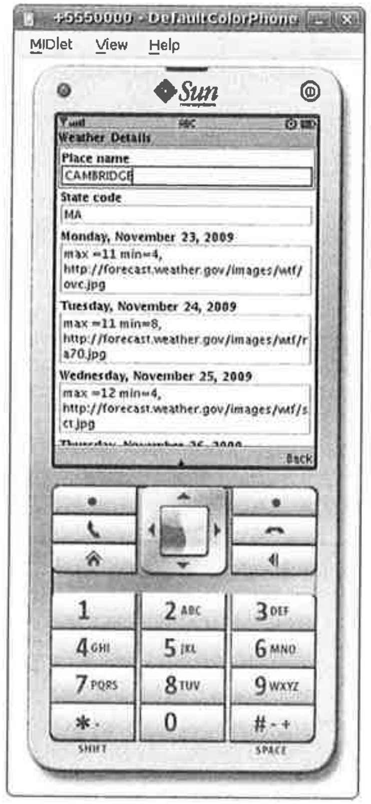
Figure 1.9 Example of using the REST Describe tool to create Web Application Description Language.



```
public static String doGetCall(String urlString) {
  String response = "";
  try {
    URL url = new URL(urlString);
    HttpURLConnection connection =
      (HttpURLConnection) url.openConnection();
    if ([illegible].length() > 0 && [illegible].length() > 0) {
      String userPassword = [illegible]
      String encoding =
        new sun.misc.BASE64Encoder().encode(userPassword.getBytes());
      connection.setRequestProperty([illegible] + encoding);
    }
    connection.setRequestMethod([illegible]);
```

```
public static void main(String args[]) {
    Geocode geocode = new Geocode([illegible]
    String result = geocode.submit();
    [illegible]
```

Figure 1.10 Example of code generated from Web Application Description Language using the REST Describe (visualized in Netbeans).

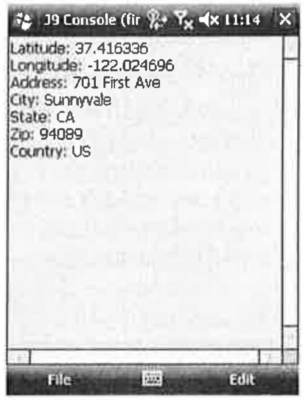


Weather Details
Place name
CAMBRIDGE
State code
MA
Monday, November 23, 2009
max =11 min=4,
http://forecast.weather.gov/images/wtf/ovc.jpg
Tuesday, November 24, 2009
max =11 min=8,
http://forecast.weather.gov/images/wtf/ra70.jpg
Wednesday, November 25, 2009
max =12 min=4,
http://forecast.weather.gov/images/wtf/sct.jpg

Figure 1.11 A mobile Web service consumer for the `WeatherForecast` service in an emulator.

Apache Muse, the developer can generate service skeletons from WSDL files and write his/her code based on OSGi containers. Specific tools like RESCUE allow Web services to be written like a plain Java object. However, with RESCUE, the service code has to be executed within a particular hosting environment. Listing 1.4 shows a simplified example of a Web service based on RESCUE for a medical staff that accepts reports about victims (operation `addNewVictim`) and that returns the profile of the staff (operation `getProfile`).

Most of the tasks involved to service discovery, call-back, security, and identity management have to be implemented by the developer. Currently, there is a lack of supporting code generation for these complex tasks. Thus, high-level libraries and tool assistance are needed to help the developer simplify these tasks. Listings 1.5 and 1.6 give an example of using the jSLP tool (http://jslp.sourceforge.net), a library



```
Latitude: 37.416336
Longitude: -122.024696
Address: 701 First Ave
City: Sunnyvale
State: CA
Zip: 94089
Country: US
```

Figure 1.12 A mobile Web service consumer for the Yahoo! Geocoding service in an HTC Tytn II.

```
//....
import at.ac.tuwien.vitalab.middleware.AWebService;
public class MedicalStaffService extends AWebService {
  public MedicalStaffService() {
  super("MedicalStaffService","http://www.infosys.tuwien.ac.
at/medicalstaffservice");
  }
   public void onDeploy() {
     //...
  }
  public void onUndeploy() {
  //..
  }
  public int addNewVictim(String location, String
victimStatus) {
    //..
    return myCurrentLoad ;
  }
  public String getProfile(String typeOfStaff) {
    //...
    return myProfile ;
  }
//...
}
```

Listing 1.4 Simplified example of a Web service in RESCUE.

```
 //...
Advertiser advertiser = ServiceLocationManager.
getAdvertiser(new Locale("en"));
ServiceURL myService = new
ServiceURL("service:disastersupport:
http://192.168.218.101:8080/services", 3600);
Hashtable attributes = new Hashtable();
attributes.put("ServiceID","MedicalStaffService")
attributes.put("Team","MedicalTeam");
attributes.put("TypeOfStaff","Nurse");
//...
advertiser.register(myService, attributes);
```

Listing 1.5 Simplified example of publishing mobile services with jSLP.

```
 Locator locator = ServiceLocationManager.getLocator(new
Locale("en"));
// find all disaster support services belonging to medical
staff
ServiceLocationEnumeration sle = locator.findServices(new
ServiceType("service:disastersupport"), null,
"(Team=MedicalTeam)");
while (sle.hasMoreElements()) {
   ServiceURL serviceURL = (ServiceURL)sle.nextElement();
   //...
}
//..
```

Listing 1.6 Simplified example of discovering mobile services using jSLP.

implementing SLP, to publish and discover the `MedicalStaffService` above. In this example, we assume that there are several medical staff available in a large-scale disaster scenario and each staff has the `MedicalStaffService` in his/her PDA. The service publishes its hosting environment information and other meta-data while the consumer would like to search only services on PDAs of nurses. Of course, the assumption is that the consumer knows how to invoke the service.

1.5 Real-World SOA Mobile Services

To demonstrate the benefit of SOC/SOA, this section presents a real-world example of mobile Web services. This example is built on our experiences from our research projects in collaborative working environments (CWEs). Modern CWEs introduce the need of various real-world SOA-based mobile Web services because today's team-work is performed by dynamic teams that are established on demand, use various mobile devices, and need to access a vast source of data and services in order to fulfill their tasks.

Our example discusses SOAP-based mobile services and consumers for disaster responses performed in the EU WORKPAD project (http://www.workpad-project.eu). The WORKPAD system [12] supports disaster management on the basis of two main ideas: (i) multiple supporting teams working on the field using mobile devices to collect information about the disaster, and (ii) teams accessing vast sources of information from different organizations to optimize their tasks and store the collected disaster information to organizational information systems. In doing so, each member uses a mobile device, and members need to interact with each other.

SOAP-based mobile Web services and consumers have been widely used in WORKPAD tools, as shown in Figure 1.13. Mobile devices are used in *ad hoc* and team collaborations where dedicated infrastructures are not available. Such collaborations normally require flexible and interoperable services while running on mobile devices and being integrated with various other services. Therefore, WORKPAD supports mobile Web services for managing context information and

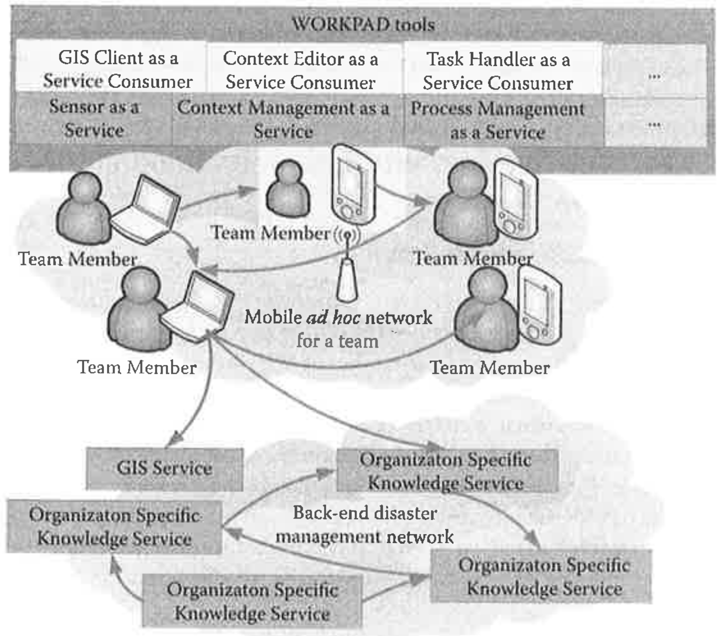

WORKPAD tools
GIS Client as a Service Consumer
Context Editor as a Service Consumer
Task Handler as a Service Consumer
...
Sensor as a Service
Context Management as a Service
Process Management as a Service
...
Team Member
Team Member
Team Member
Team Member
Team Member
Mobile *ad hoc* network for a team
GIS Service
Organizaton Specific Knowledge Service
Organizaton Specific Knowledge Service
Back-end disaster management network
Organizaton Specific Knowledge Service
Organizaton Specific Knowledge Service

Figure 1.13 (Simplified and abstract) Simple Object Access Protocol mobile services in the WORKPAD project.

for coordinating collaborative tasks. The context management system of WORKPAD [5,27] provides various types of context information to different clients such as GIS-based mobile applications and task management. Context sensors and context management services are mobile Web services and context information is described in XML, thus facilitating the integration between the context management system and other applications. Task management provides a SOAP-based BPEL engine for coordinating tasks [47]. Furthermore, GIS-based mobile applications utilize Web service technologies to access GIS and other information from the back-end services that are also Web services.

Without using Web services, WORKPAD would face a great challenge in the integration of different sources of data. For example, GIS data are now provided through Web services (e.g., see Reference [48]). Thus, by developing SOAP-based GIS clients, WORKPAD can ensure that its GIS software is interoperable and reusable. Various information sources from different government agencies are hard to access through mobile applications, if these sources are not wrapped into Web services. If the context management system was not based on Web services solution, it would be hard to not only integrate the context information to different clients but also to share the context information among different teams. Furthermore, without using Web services, WORKPAD components would be a very specific implementation that would not be highly reused in other works. For example, the Web service-based context management system can be used in SOA-based smart-home environments [49].

1.6 SOA/Web Services and Other Technologies/Styles for Mobile Services

1.6.1 OMA/Mobile Web Services

The Open Mobile Alliance (OMA) (http://www.openmobilealliance.org) has defined the OMA mobile Web service (MWS) to provide guidelines and specifications for the integration and interoperability of Web services with the OMA architecture. With respect to architectural styles, MWS supports SOAP-based models. MWS considers cases when Web service consumers support and do not support full Web service stacks. Thus, both cases in Figure 1.3 are included: the interaction between mobile Web service consumers and mobile Web services can be direct or through proxy or intermediaries. With respect to service discovery, MWS supports centralized discovery based on UDDI and provides security guidelines for both transport and message security based on SSL/TSL and WS-Security.

1.6.2 CDC/OSGi Model

Another trend to support SOA-based mobile services is to use the OSGi technology defined by the OSGi Alliance (http://www.osgi.org). The OSGi technology specifies

a dynamic component system for Java. In this system, components interact with each other within a single system. OSGi components can be published, discovered, and composed dynamically at runtime. They can also be deployed and undeployed on demand and be exposed as Web services. Various OSGi implementations have been provided for different platforms, including mobile devices.

CDC/OSGi is a capable platform for developing mobile services based on SOA. Various platforms are supported by CDC/OSGi. Examples are Sprint Titan (http://developer.sprint.com/site/global/develop/technologies/sprint_titan/p_sprint_titan.jsp) and Apache MUSE (http://ws.apache.org/muse/) that provides development of Web services atop OSGi. R-OSGi [50] is a middleware built atop OSGi that transparently connects, deploys, and executes OSGi services spanning multiple OSGi containers; thus, it can be used for developing P2P and collaborative mobile Web services.

1.6.3 Event-Driven Architecture and SOA

The event-driven architecture (EDA) [51] is a computing paradigm in which changes are sensed and captured in events and corresponding actions are performed according to events. The basic tenet of EDA is that an event reflects a significant change of something by gathering meaningful information about the change. The fundamental conceptual entities of an EDA system are event sources (event emitters) to generate events and event sinks (event consumers) to perform actions based on events (e.g., filtering, relay, or processing events). Events are sensed, transferred, and processed through a system of event channels loosely coupling distributed event sources and sinks.

EDA systems do not necessarily include SOA-based services. However, the need to integrate events from disparate sources has fostered the integration between EDA and SOA [52,53]. This integration has been proposed through standard bodies as well as architecture solutions. WSEventing (http://www.w3.org/Submission/WS-Eventing/) and WS-ECA [54] have been proposed for supporting event subscription, propagation, and event–condition–action rules.

1.6.4 Web Applications, WAP, and SIP

Web applications, WAP, and SIP models are other technologies/architecture styles that are related to SOA and mobile services (see also Section 1.3.1.3 and Figure 1.4). Although they are different technologies/styles, all of them require proxy in order for the mobile applications on mobile devices to interact with other services at the back end.

The Web application model for mobile services is widely used in practice. Conceptually, the Web application model follows a three-tier architecture: the mobile service consumer is responsible for the presentation layer; the Web application, the application tier, acts as a gateway to provide dynamic Web content to the mobile service consumer; and the mobile services provide data and content to the gateway. The mobile service consumer is normally developed based on Web forms, XForm, JavaScript, and HTML/XHTML, whereas the Web application gateway uses dynamic Web content technologies such as ASP and JSP. Typically, the Web application model widely supports synchronous interactions in which the requester pushes the requests and receives the content. Recently, advanced AJAX techniques for mobile devices also support the requester to asynchronously poll results. Some toolkits are available for mobile AJAX, such as MobileWeb (http://mymobileweb.morfeo-project.org/). A single-purpose mobile Web application can also be packaged for download and installation on mobile devices; this is called a mobile widget [55]. This model is quite suitable for applications that require Web contents. Examples of Web application model toolkits are the Yahoo! Blueprint Platform, Microsoft ASP.NET mobile, and Microsoft Mobile Internet Toolkit.

Other architectural styles for mobile services are based on WAP (http://www.wapforum.org/what/technical_1_2_1.htm) and SIP (http://www.ietf.org/html.charters/sip-charter.html). WAP defines how an application on mobile phones or PDAs can access the Internet through HTTP. WAP defines a protocol stack atop different wireless data networks, such as SMS, GPRS, and UMTS. This protocol stack includes protocols dealing with datagram, transport, transaction, session control, and environments. A WAP application accesses the Internet through a WAP gateway that acts as a proxy. WAP has been widely used in services, such as mobile mails for Mobile CRM [56], transportation information systems (http://www.tfl.gov.uk/tfl/livetravelnews/mobileservices/wap/default.asp), and mobile banking. SIP is a signaling protocol widely used in multimedia mobile services, such as streaming multimedia delivery, instant messaging, and presence service. As it focuses on multimedia aspect, using SIP alone will support only a few types of mobile services. However, SIP can be combined with SOA-based services to provide integrated communication services and enterprising business services. This type of application is particularly important for the mobile user in networked enterprises and collaborative environments. Examples of SIP together SOA are IMS [21] and Akogrimo [11,57]. A SIP for mobile devices is also developed in Java ME (JSR 180—Session Initiation Protocol for J2ME 1.0.1).

1.7 Challenges for Future Research

As partially discussed in previous sections, we need to develop further techniques to bring the full advantage of SOA-based solutions to mobile services.

1.7.1 Supporting Mobile Web Services on Mobile Devices

Supporting tools should not only focus on the client side but also on mobile Web services in mobile devices. This will require great research effort for the development of service management systems that have to take into account service mobility, runtime deployment, and service continuity on mobile devices. Here, ensuring correct information about services and service continuity is a great challenge. The issue related to service registration and discovery when Web services are hosted in mobile devices, as discussed in Reference [6], is still valid. We need to combine service migration and adaptation with service monitoring and management techniques. This issue is also related to service disruption due to failure of devices and networks.

1.7.2 Supporting REST Mobile Services

Important supports of REST-based mobile services, such as service discovery and resource management, should be provided. In particular, for publishing and discovering REST-based Web services a new set of protocols has to be developed.

1.7.3 Programming Supporting Tools

Although various programming environments are provided, the developer lacks useful tools to debug, profile, and analyze their mobile services. In particular, given the strong connection between performance and energy consumption in mobile devices, tools for analyzing energy consumption and performance are particularly interesting. Currently, only a few works have been devoted for this research issue.

1.7.4 Context Sharing in Mobile Services

When mobile services are integrated from different providers, sharing context information among these services is challenging. Context sharing is a key to reduce user intervention and improve service provisioning and adaptation. Existing standards have proposed sharing user identity but it is not enough as the context associated with a usage is much more than identity. Thus, standard protocols for specifying, propagating, and managing context in a federation of services are desired. This research is also involved with context-aware computing domains.

1.7.5 User-Defined Opportunistic Composition and Creation of Mobile Service

When a lot of mobile services are available, the user will have many opportunities to compose mobile services on their own. SOA solutions have helped to simplify service discovery processes and many service composition techniques have been proposed, yet they are complex and assume advanced knowledge. How can we support a novice user to compose mobile services in a place where, by chance, his/her mobile device detects many interesting mobile services provided by other users and onsite service providers? In addition, how can we support the user to define his/her own mobile services? Although some initial ideas have been proposed [36,58], these are premature.

1.8 Conclusion

Over the last few years, the increasing mobility of the user and the advanced, powerful mobile networks and devices have fostered emerging mobile services for different scenarios. Mobile services are no longer just for accessing Web contents and for personal use but also for enterprise use and collaborative work. Furthermore, there is no longer a clear boundary between mobile services hosted in mobile networks and on the Internet. This requires mobile applications to access vast information and services from anywhere with any device. As a result, SOC/SOA technologies, in particular, Web service technologies, have been utilized in mobile services, solving many integration problems.

In this chapter, we analyze state-of-the-art SOC/SOA for mobile services with a focus on mobile Web services. Both SOAP-based and REST-based architectural styles have been discussed. On the one hand, many Web service solutions, based on SOAP and REST technologies, have been introduced for mobile services, enhancing substantially how mobile services should provide and interact with the mobile user. On the other hand, Web service solutions for collaborative works, distributed health care, and enterprises have been developed, extending the use of mobile services for individual need to team work and enterprise business. Both REST and SOAP mobile Web services have different capabilities and this difference needs to be analyzed to select the right architectural style. Other chapters in this book provide further state-of-the-art research on particular topics that are sketched in this chapter, such as security, service discovery, monitoring, and performance analysis.

1.9 Further Reading

Several books have presented many fundamental concepts of SOC/SOA, such as *Web Services: Concepts, Architecture and Applications* [59] and *SOA Principles of Service Design* [60]. With respect to the development of mobile services, there are valuable References for further reading. *Mobile Web Services: Architecture and Implementation* [61] presents main concepts in designing Web services for mobile systems. Generic concepts such as addressing, service discovery, identity management, and security in SOA-based environments are covered. This book also presents implementation techniques and examples of mobile Web services. However, it focuses on SOA based on Nokia service development APIs for S60 platform.

Mobile Web Services [62] presents main technologies for implementing wireless mobile services with a focus on mobile Web services providing Web contents.

Among others, it covers very well WAP, wireless content representations, location management, privacy, and mobile user context. However, it does not focus on SOA solutions for mobile Web services.

Enabling Technologies for Mobile Services: The MobiLife Book [3] presents a compelling landscape about mobile services based on the user-centered design process. It analyzes requirements of the mobile world and discusses the mobile services architecture accordingly. Key enabling technologies such as context management framework, multimodal and personalization technologies, and trust and privacy are well presented. This book also presents Reference applications, best practices, and marker analysis of mobile services. It is an excellent reading source for understanding mobile applications and services.

Mobile service-oriented architectures in general are discussed in Reference [63]. This discussion does not indicate any specific Web services platform. Some mobile programming toolkits were evaluated in Reference [64]. This evaluation discussed the performance of gSOAP and kSOAP Web service consumers in embedded devices.

This chapter gives an overview of the main techniques for mobile Web services with a focus on programming support for service development, service invocation, message handling, and discovery. Therefore, it does not present these techniques in detail in terms of qualitative and behavioral analysis, such as performance, scalability, energy consumption, and detailed protocol structures. Such details can be found in external References, other chapters of this book, and the above-mentioned further readings.

EXERCISES

1. Survey existing mobile services, such as Google and StrikeIron services, and map them into a table of direct/indirect interaction, SOAP, REST, and other architectural styles; and compare their strengths and weaknesses.
2. Not all published WSDL files of Web services are compatible with Java ME. Analyze existing WSDL files, find common incompatible properties, and present your suggestion.
3. Assume that some services are hosted in a mobile device and services are described by WSDL and UDDI. Analyze possible issues that arise when the service provider migrates the services to another hosting environment. What should be done to ensure that the service discovery process is not corrupted?
4. Analyze possible service discovery protocols for REST-based Web services. Implement a P2P-based service discovery for REST Web services.
5. Compare data transfer using JSON and XML format. Which format is suitable for which application types? Perform a performance comparison for similar Web services that offer both JSON and XML (e.g., Google and Yahoo! mobile services).

6. Currently, many REST-based Web services return XML and/or JSON, but programming toolkits assume the developer will utilize specific libraries (e.g., kXML or JSONObject) to parse the returned data. Develop a small project to integrate tools for generating code from WADL and tools for parsing XML/JSON data for REST clients.
7. Study service information associated with mobile Web services on mobile devices. Compare the importance of the service information of mobile Web services with that of Web services in other platforms in terms of service discovery and service management.
8. Analyze the dependency between mobile Web services and the contemporary operating systems for mobile devices (PDAs and smart phones) in terms of concurrency processing and service disruption. Analyze the impact of issues on the selection of architectural styles, service invocation models, and fault management.

Acknowledgments

The effort spent in writing this chapter is partially supported by the European Union through the FP6 projects inContext and WORKPAD, and the FP7 project SM4All. The research in this chapter is partially a result of our experiences gained from the development of SOA-based solutions for teamwork, disaster management, and smart environments in the inContext (http://www.in-context.eu), the WORKPAD (http://www.workpad-project.eu), and the SM4All (http://www.sm4all-project.eu) projects. We are grateful to all our colleagues in inContext, WORKPAD, and SM4All for fruitful discussion and sharing ideas on the development of mobile Web services. Our discussions on Vimoware/RESCUE and the WORKPAD example are based on research results conducted within the WORKPAD project.

References

1. Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, 2007.
2. Ivar Jørstad, Schahram Dustdar, and Do Van Thanh. An analysis of current mobile services and enabling technologies. *IJAHUC*, 1(1/2):92–102, 2005.
3. Mika Klementtinen, editor. *Enabling Technologies for Mobile Services: The MobiLife Book*. West Sussex, England: John Wiley & Sons, October 2007.
4. Ivar Jørstad, Schahram Dustdar, and Do Van Thanh. Service-oriented architectures and mobile services. In Jaelson Castro and Ernest Teniente, editors, *CAiSE Workshops (2)*, pp. 617–631. Porto: FEUP Edicoes, 2005.

5. Hong Linh Truong, Lukasz Juszczyk, Shariq Bashir, Atif Manzoor, and Schahram Dustdar. Vimoware—a toolkit for mobile web services and collaborative computing. In *SEAA '08: Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 366–373, Parma, Italy, September 3–5, 2008.
6. Stefan Berger, Scott McFaddin, Chandra Narayanaswami, and Mandayam Raghunath. Web services on mobile devices—implementation and experience. *wmcsa*, 0:100, 2003.
7. Ari Shapiro and Andreas Frank. Mobile SOA: End-to-end Java^{TM} technology-based framework for network services. In *Proceedings of JavaOne 2008 Conference*, San Francisco, May 5–9, 2008.
8. Mobile world celebrates four billion connections, http://www.gsmworld.com/newsroom/press-releases/2009/2521.htm. Last accessed: February 26, 2009.
9. Gerd Andersson, Adrian Bullock, Jarmo Laaksolahti, Stina Nylander, Fredrik Olsson, LinderMarie Sjö, Annika Waern, and Magnus Boman. Classifying mobile services. *SICS Technical Report T2004:04*. Swedish Institute of Computer Science ISSN 1100-3154, 2004. http://eprints.sics.se/2349/01/SICS-T–2004-04–SE.pdf
10. Marco Savini, Andreea Ionas, Andreas Meier, Ciprian Pop, and Henrik Stormer. The eSana framework: Mobile services in eHealth using SOA. In *Proceedings of the Second European Conference on Mobile Government*, Brighton, August 30–31 and September 1, 2006, ISBN: 0-9763341-1-9.
11. The Akigrimo project. http://www.mobilegrids.org/
12. Tiziana Catarci, Massimilano de Leoni, Andrea Marrella, Massimo Mecella, Berardino Salvatore, Guido Vetere, Schahram Dustdar, Lukasz Juszczyk, Atif Manzoor, and Hong-Linh Truong. Pervasive software environments for supporting disaster responses. *IEEE Internet Computing*, 12(1):26–37, 2008.
13. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture (W3C Working Group note, February 11, 2004), 2004. http://www.w3.org/TR/ws-arch/. Last accessed: February 10, 2009.
14. Gregory Hackmann, Mart Haitjema, Christopher D. Gill, and Gruia-Catalin Roman. Sliver: A BPEL workflow process execution engine for mobile devices. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pp. 503–508. The Netherlands: Springer, 2006.
15. A. van Halteren and P. Pawar. Mobile service platform: A middleware for nomadic mobile service provisioning. In *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006)*, pp. 292–299, Montreal, Quebec, June 19–21, 2006.
16. Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine, 2000.
17. S. McFaddin, D. Coffman, J.H. Han, H.K. Jang, J.H. Kim, J.K. Lee, M.C. Lee et al., Modeling and managing mobile commerce spaces using restful data services. In *9th International Conference on Mobile Data Management, 2008, MDM '08*, pp.81–89, Berlin, April 27–30, 2008.
18. Sami Mäkeläinen and Timo Alakoski. Fixed-mobile hybrid mashups: Applying the rest principles to mobile-specific resources. In *WISE '08: Proceedings of the 2008 International Workshops on Web Information Systems Engineering*, pp. 172–182. Berlin/Heidelberg: Springer-Verlag, 2008.

19. ArcGIS mobile blog. http://blogs.esri.com/Dev/blogs/mobilecentral/archive/2008/10/20/The-Mobile-Web.aspx. Last accessed: March 13, 2009.
20. Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big"' web services: Making the right architectural decision. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *17th International World Wide Web Conference (WWW2008)*, pp. 805–814. Beijing, China: ACM, April 21–25, 2008.
21. Hechmi Khlifi and Jean-Charles Grégoire. IMS application servers: Roles, requirements, and implementation technologies. *IEEE Internet Computing*, 12(3):40–51, 2008.
22. XMLP scenarios. http://www.w3.org/TR/xmlp-scenarios/. Last accessed: March 12, 2009.
23. Guido Gehlen and Georgios Mavromatis. Mobile web service based middleware for context-aware applications. In *Proceedings of the 11th European Wireless Conference 2005*, Vol. 2, pp. 784–790. Nicosia, Cyprus: VDE Verlag, April 2005.
24. Valerie Issarny, Daniele Sacchetti, Ferda Tartanoglu, Françoise Sailhan, Rafik Chibout, Nicole Levy, and Angel Talamona. Developing ambient intelligence systems: A solution based on web services. *Automated Software Engineering*, 12(1): 101–137, 2005.
25. Fahad Aijaz, Bilal Hameed, and Bernhard Walke. Asynchronous mobile web services: Concept and architecture. In *Proceedings of 2008 IEEE 8th International Conference on Computer and Information Technology*, p. 6, Sydney, Australia, July 2008.
26. Yeon-Seok Kim and Kyong-Ho Lee. A light-weight framework for hosting web services on mobile devices. In *ECOWS '07: Proceedings of the Fifth European Conference on Web Services*, pp. 255–263. Washington, DC, USA: IEEE Computer Society, 2007.
27. Hong Linh Truong, Lukasz Juszczyk, Atif Manzoor, and Schahram Dustdar. Escape—An adaptive framework for managing and providing context information in emergency situations. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *EuroSSC*, volume 4793 of *Lecture Notes in Computer Science*, pp. 207–222. The Netherlands: Springer, 2007.
28. Martin Treiber and Schahram Dustdar. Active web service registries. *IEEE Internet Computing*, 11(5):66–71, 2007.
29. Marc Hadley. Web application description language (WADL). *Technical Report TR-2006-153*, Sun Microsystems, April 2006.
30. Robert Steele, Khaled Khankan, and Tharam Dillon. Mobile web services discovery and invocation through auto-generation of abstract multimodal interface. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)—Volume II*, pp. 35–41. Washington, DC, USA: IEEE Computer Society, 2005.
31. Satish Narayana Srirama, Matthias Jarke, and Wolfgang Prinz. Mobile web services mediation framework. In *MW4SOC '07: Proceedings of the 2nd Workshop on Middleware for Service Oriented Computing*, pp. 6–11. New York, NY, USA: ACM, 2007.
32. Satish Narayana Srirama, Matthias Jarke, Hongyan Zhu, and Wolfgang Prinz. Scalable mobile web service discovery in peer to peer networks. In *Third International Conference on Internet and Web Applications and Services, 2008, ICIW '08*, pp. 668–674. Athens, Greece: IEEE Computer Society, June 8–13, 2008.

33. Lukasz Juszczyk and Schahram Dustdar. A middleware for service-oriented communication in mobile disaster response environments. In Sotirios Terzis, editor, *MPAC*, pp. 37–42. New York, NY, USA: ACM, 2008.
34. RESCUE—Service oriented middleware for mobile devices, http://www.infosys.tuwien.ac.at/prototyp/Rescue/Rescue_index.html. Last accessed: February 26, 2009.
35. Lasse Pajunen and Suresh Chande. Developing workflow engine for mobile devices. In *EDOC '07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, p. 279. Washington, DC, USA: IEEE Computer Society, 2007.
36. Marco Pistore, Paolo Traverso, Massimo Paolucci, and Matthias Wagner. From software services to a future Internet of services. *Future Internet Assembly* 183–192, 2009.
37. Mia Tian, Thiemo Voigt, Tomasz Naumowicz, Hartmut Ritter, and Jochen Schiller. Performance considerations for mobile web services. *Computer Communications Journal*, 27(11):1097–1105, 2004.
38. Ivan Jorstad, Do van Thanh, and Schahram Dustdar. A service continuity layer for mobile services. *2005 IEEE Wireless Communications and Networking Conference*, 4:2300–2305, 2005.
39. Christoph Dorn and Schahram Dustdar. Achieving web service continuity in ubiquitous mobile networks: The SRR-WS framework. In Moira C. Norrie, Schahram Dustdar, and Harald Gall, editors, *UMICS*, volume 242 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2006.
40. kSOAP2, http://ksoap2.sourceforge.net/. Last accessed: February 26, 2009.
41. Robert van Engelen and Kyle Gallivan. The gSOAP toolkit for web services and peer-to-peer computing networks. In *CCGRID*, pp. 128–135. Washington, DC, USA: IEEE Computer Society, 2002.
42. Vittorio Miori, Luca Tarrini, and Rolando Bianchi Bandinelli. Deliverable d2.2: Requirements analysis for footprint and power constrained devices; light—xml-innovative generation for home networking technologies. *Technical Report*, CNR, Pisa, Italy, May 2005. http://dienst.isti.cnr.it/Dienst/Repository/2.0/Body/ercim.cnr.isti/2005-PR-02/pdf?tiposearch=ercim&langver=
43. Nokia web services platform. http://www.forum.nokia.com/Resources_and_Information/Explore/Other/Web_Services/. Last accessed: March 13, 2009.
44. Peter Braun and Ronny Eckhaus. Experiences on model-driven software development for mobile applications. In *ECBS '08: Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pp. 490–493. Washington, DC, USA: IEEE Computer Society, 2008.
45. Juergen Dunkel and Ralf Bruns. Model-driven architecture for mobile applications. In Witold Abramowicz, editor, *Business Information Systems, 10th International Conference, BIS 2007*, Poznan, Poland, April, pp. 470–483. Berlin/Heidelberg: Springer-Verlag, 2007.
46. Mariano Belaunde and Paolo Falcarin. Realizing an MDA and SOA marriage for the development of mobile services. In *ECMDA-FA '08: Proceedings of the 4th European Conference on Model Driven Architecture*, pp. 393–405. Berlin/Heidelberg: Springer-Verlag, 2008.
47. Daniele Battista, Massimiliano de Leoni, Alessio De Gaetanis, Massimo Mecella, Alessandro Pezzullo, Alessandro Russo, and Costantino Saponaro. Rome4eu: A web service-based process-aware system for smart devices. In Athman Bouguettaya, Ingolf

Krüger, and Tiziana Margaria, editors, *ICSOC*, volume 5364 of *Lecture Notes in Computer Science*, pp. 726–727. The Netherlands: Springer, 2008.
48. Shengru Tu and Mahdi Abdelguerfi. Web services for geographic information systems. *IEEE Internet Computing*, 10(5):13–15, 2006.
49. Marco Aiello and Schahram Dustdar. Are our homes ready for services? A domotic infrastructure based on the web service stack. *Pervasive and Mobile Computing*, 4(4):506–525, 2008.
50. Jan S. Rellermeyer, Gustavo Alonso, and Timothy Roscoe. R-OSGi: Distributed applications through software modularization. In Renato Cerqueira and Roy H. Campbell, editors, *Middleware*, volume 4834 of *Lecture Notes in Computer Science*, pp. 1–20. The Netherlands: Springer, 2007.
51. Brenda M. Michelson. Event-driven architecture overview. *Patricia Seybold Group*, February 2006. http://soa.omg.org/Uploaded%20Docs/EDA/bda2-2-06cc.pdf
52. Jack van Hoof. How EDA extends SOA and why it is important, September 2006. http://soa-eda.blogspot.com/2006/11/how-eda-extends-soa-and-why-it-is.html
53. Jeff Hanson. Event-driven services in SOA, January 2005. http://www.javaworld.com/javaworld/jw-01-2005/jw-0131-soa.html
54. Jae-Yoon Jung, Jonghun Park, Seung-Kyun Han, and Kangchan Lee. An ECA-based framework for decentralized coordination of ubiquitous web services. *Information & Software Technology*, 49(11–12):1141–1161, 2007.
55. Marcos Caceres. Widgets 1.0: Packaging and configuration—W3C working draft, December 22, 2008. http://www.w3.org/TR/2008/WD-widgets-20081222/
56. Kevin H.W. Shen and Daniel C.H. Lee. WAP mail service and short message service for mobile CRM. In *MSE '00: Proceedings of the 2000 International Conference on Microelectronic Systems Education*, p. 201. Washington, DC, USA: IEEE Computer Society, 2000.
57. Juergen M. Jaehnert, Stefan Wesner, and Victor A. Villagra. The Akogrimo mobile grid Ref. architecture—Overview. http://www.mobilegrids.org/modules.php?name=UpDownload&req=getit&lid=108, November 2006, whitepage.
58. Daniel Schall, Hong Linh Truong, and Schahram Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, 2008.
59. Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architecture and Applications*. Berlin/Heidelberg: Springer-Verlag, 2004.
60. Thomas Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.
61. Frederick Hirsch, John Kemp, and Jani Ilkaka. *Mobile Web Services: Architecture and Implementation*. New York: John Wiley & Sons, 2006.
62. Ariel Pashtan. *Mobile Web Services*. New York, NY, USA: Cambridge University Press, 2005.
63. Jilles van Gurp, Anssi Karhinen, and Jan Bosch. Mobile service oriented architectures (MOSOA). In Frank Eliassen and Alberto Montresor, editors, *DAIS*, volume 4025 of *Lecture Notes in Computer Science*, pp. 1–15. The Netherlands: Springer, 2006.
64. Daniel Schall, Marco Aiello, and Schahram Dustdar. Web services on embedded devices. *IJWIS*, 2(1):45–50, 2006.