Web Service Composition on Smartphones: The Challenges and a Survey of Solutions

Gebremariam Mesfin[1], Gheorghita Ghinea[1], Tor-Morten Grønli[2(✉)], and Muhammad Younas[3]

[1] Department of Computer Science, Brunel University, London, UK
{gebremariam.assres,george.ghinea}@brunel.ac.uk
[2] Mobile Technology Laboratory, Department of Technology, Westerdals Oslo ACT, Oslo, Norway
tmg@westerdals.no
[3] School of Engineering, Computing and Mathematics, Oxford Brookes University, Oxford, UK
m.younas@brookes.ac.uk

Abstract. Today, smartphones are capable of hosting a large variety of applications as clients for the classical as well as service-based business applications. They can also be envisaged for composing and hosting service-based thick client applications aiming at exploiting users' creativity and resolving mobile connectivity challenges, respectively. However, the challenges in input mechanisms, storage, and screen size of smartphones all limit the operations of Web service composition. These challenges can be addressed through a criteria-based selection process of appropriate Web service and associated technologies. Accordingly, the REST services, semi-automatic service composition with Web 2.0 technologies (HTML5, and JavaScript APIs), JSON-based messaging and data serialization format, as well as the cross-platform mobile client application development approach are found more suitable for composing Web services on the constrained smartphone. All together, they constitute a stack of appropriate technologies to implement resource-oriented architecture on the smartphone.

Keywords: Web service · REST · SOAP · Mobile · Smartphone · JSON · Web 2.0 HTML5 · JavaScript APIs · Cross-platform · Composition techniques Semi-automatic · SOA · ROA

1 Introduction

The smartphone landscape has made significant leap and transformed the phone from a handset used merely for voice calls into a state-of-the-art mobile computing device [1]. Today, smartphones are capable of hosting a large variety of client-side applications in areas such as banking, education, and health. There are also attempts to turn the smartphone into a development environment such as the Android IDE[1].

[1] https://www.android-ide.com/.

M. Younas et al. (Eds.): MobiWIS 2018, LNCS 10995, pp. 126–141, 2018.
https://doi.org/10.1007/978-3-319-97163-6_11

On the other hand, developing applications using service-oriented architecture (SOA) and resource-oriented architecture (ROA) has paramount importance [2]. In this regard, client-side applications which are composed of SOA/ROA based Web service components can also be hosted on the smartphones. However, to the best of our knowledge, the smartphone is not yet in use as a Web service composition device.

The end-user development research community, on the other hand, confirms that end-users can innovatively develop their own applications with little or no programming knowledge [3, 4]. Thus, a possibility of composing Web services on smartphones would enable the exploitation of smartphone users' creativity so that they can develop their own applications and also contribute their work to app stores for use by others. For example, a supplier may need a custom made application for his smartphone to find customers with the best price offer for products to sale, negotiate a contract, arrange billing details, and logistics for shipment as shown in Fig. 1.

Figure 1 depicts a system design (a scenario) of Web service composition on a smartphone. The figure demonstrates Web services being provided from customers' devices (left), the smartphone being used in the wiring of workflows for composing Web services (middle), and Web services being provided from third-party organizations' devices (right). Accordingly, the supplier performs Web service composition by wiring interfaces of the exposed services.

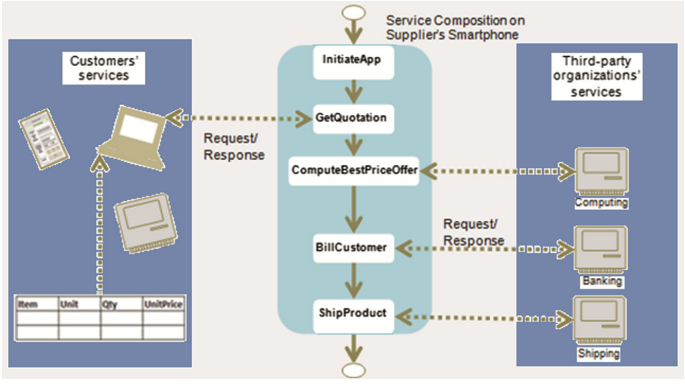

Fig. 1. Service composition on smartphones

The above described scenario requires identifying the most appropriate Web service and associated technologies for simplicity of composing on smartphones. In this regard, although many of the available mash-up tools are claimed to be for end-users too, they are generally targeted towards data integrators for use on classic computing devices. Moreover, practice shows that they also require significant programming skill [5].

Additionally, a client application is required to be thick enough for interactive business applications [6]. It is also worth mentioning the situations of low bandwidth and intermittent Internet connectivity like the case of many developing economies [7]. Thus, hosting composite applications on the smartphone as a thick-client would compensate the negative impact of intermittent and low bandwidth mobile connectivity. However,

to the best of our knowledge, there is no established trend in the hosting of service-based (composite) client-side applications on the smartphone.

Thus, composing Web services directly on the smartphone and hosting the resulting client-side application on it has paramount importance. However, smartphones are still constrained in terms of input mechanisms, storage, and screen size and these are essentially the challenges which could limit the operations of Web service composition and hosting.

In fact, service composition is an existing challenge already [8]. For example, Beaton et al. [9] reveal that service consumers encounter challenges such as fragmentation of the discovery user experience across multiple interfaces, confusing hierarchies of service navigation, lack of business modeling support, hidden relationships between services, and inconsistent Web service interface design. Some of the manifestations of these challenges are through the difficulties developers encounter while assembling data structure in Web service parameters, cycles of errors due to unclear control parameters in data structures, and difficulties to understand long identifier names.

Thus, the fast proliferation of Web services and dynamically changing user requirements are demanding increasing design consistency and simplicity of Web services. In addition, addressing design issues for the use of Web services in smartphone applications development challenges even more [8].

In smartphone application development, the software design process is driven by the limited resource of the specific device. In this paper, we concentrate on and examine Web service technologies for the required resources for composing and hosting on smartphones and the degree to which the resulting smartphone application is lightweight and usable enough to please end-user developers. Thus, the small and diverse screen sizes, storage capacity, and configurations of the smartphone need to be considered when developing Web services. In addition, issues such as performance, footprint size, and usability must all be taken into account. Accordingly, we conduct a literature-based survey of available Web service and related technologies relevant to compose Web services on smartphones structured in four categories, namely:

- Types of Web services;
- Messaging and data serialization formats;
- Service composition techniques and languages; and
- Mobile client application development approaches.

In addition, we evaluate candidate technologies for their desirable features and limitations and proposed stack of appropriate technologies aimed towards tailoring Web service composition environment for a smartphone. Thus, the paper is organized as follows. In Sect. 2, we present a survey of Web service and associated technologies in each of the above mentioned categories. Sections 3, 4 and 5 respectively detail evaluation of these technologies, discussion of results, and the conclusions drawn.

2 Survey of Web Services and Associated Technologies

Web services are implementation technologies of SOA and ROA. They are inspired by the need to integrate and reuse software for the dynamically changing and highly collaborative stakeholder needs of today's businesses. Additionally, we provide the associated technologies which are used for developing applications using Web services as presented below. Accordingly, we approached the survey by classifying the Web services and associated technologies in four categories as depicted in Fig. 2.

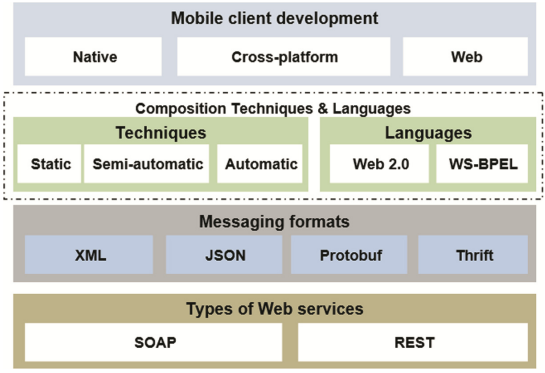

Fig. 2. Web service technologies in categories

Figure 2 shows a client-server interaction between mobile clients and Web services through messaging formats. In addition, the composition techniques and languages component in the figure signifies the technology with which the code logic of a mobile client can be developed. Accordingly, these Web services and associated technologies are described next.

2.1 Type of Web Service

Web services can be broadly categorized into SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). SOAP services work in the context of SOA based-on foundational Web technologies like the DOM, URL based discovery, and HTTP messaging with XML [10]. REST services, on the other hand, correspond to the ROA which works based-on the concept of Web resources' URL addressability and the mechanism of passing-in parameters via the URL [10, 11]. A review on the challenges of composing Web services on smartphones is presented next.

In SOA, self-contained software services are used to build applications or other software services and the process is called Web service composition. Thus, software components are made interoperable leveraging the flexibility to meet customers' use cases, to have access to data in a unique context, and to meet customers' preferences to interact from a smartphone device [12].

Srirama et al. [8] pointed out that the smartphone can be used as a Web service requestor, and as a host. However, it faces a number of hardware challenges such as the small screen size which would limit the use of the smartphone as Web service composition device [19].

The composition of Web services on smartphones could also be challenged due to usability limitations of the Web services themselves. For example, usability of Web services could be impeded due to the complex data structures in SOAP; and the manual operation in REST [9, 11]. Thus, from the perspective of type of Web service, composition requires programming skill which prohibits non-programmers for building applications [5].

2.2 Messaging Format

The type of messaging and data serialization format is among such associated technologies for Web service composition. Nurseitov et al. [15] pointed out that the choice of data format can have significant consequences on data transmission rates and performance. Today, many data formats are available like XML, JSON, Protobuf, efficient XML interchange (EXI[2]), and Thrift [17]. However, with the ever increasing access to a diversity of Internet-connected devices, choosing the appropriate data serialization format for use on the constrained smartphone has become a challenge [16]. Thus, comparing data format as text versus binary with respect to the smartphone has paramount importance.

2.3 Web Service Composition

Orchestration and choreography are among the service composition techniques applicable on both SOAP and REST Web services and selecting among them is dependent on the purpose of the application in question [18]. In this paper, we concentrate on the use of a smartphone as a coordinator in the orchestration of Web services for building applications. However, the difference in the conventional practices of the SOAP and REST services has induced differences in the composition techniques and languages.

Thus, service composition approaches can be categorized either as control flow, data flow, and assisted by service consumer [19]; or as static, automatic, and semi-automatic [20]. The composition languages, on the other hand, are BPMN, WS-BPEL and HTML5 together with JavaScript APIs; these can all be categorized as graphical modeling, XML-based, and Web 2.0 technologies, respectively. Both the BPMN and WS-BPEL have standard business process execution engines. Here our intention is not to provide generic comparison of such a varying collection of technologies and their variants like the BPEL for REST services[3] in the WS-BPEL. Accordingly, we concentrate on the WS-BPEL industry standard for its role in the composition of SOAP services [18], and HTML5 which is in use as a mash-up language together with JavaScript APIs (in the Web 2.0 context) for REST services as representative alternative technologies for our

[2] http://www.w3.org/XML/EXI/.
[3] http://ode.apache.org/.

perspective. These technologies are evaluated with respect to the requirements of composing on smartphones.

2.4 Mobile Client Application

The composition tool in use also influences the task of composing Web services on smartphones. Essentially, this represents the mobile client application and it denotes the Web service consumer in the composition. Due to the proliferation of smartphones, it is a common practice today that Internet users employ mobile Web client applications such as Opera, Chrome, and Safari browsers for their Web experiences [1]. In addition, native and cross-platform smartphone application development approaches are also employed to communicate with Web services [21]. Thus, studying the characteristics of mobile client applications with respect to the goal of composing Web services on smartphones and the need to host the composite applications is of paramount importance.

In general, the existing literature provides significant insight into the Web services and the associated technologies; and Web service composition. However, to the best of our knowledge, composition with respect to the resource limited smartphone environment is not sufficiently explored. Thus, further review of literature focusing on the influence of the technologies for enabling composition on smartphone platforms is presented next.

3 Evaluation of Technologies

3.1 Methodology

This study was conducted using literature survey on Web services and associated technologies according to the four categories depicted in Fig. 2. Performance, amount of communication and storage footprint; and the level of simplicity to work on the constrained input components and the small but variable screen real-estate of the smartphone landscape were given due consideration for evaluating each category presented in the previous section. Thus, description of each evaluation criteria is presented next.

Type of Web Service. Here, we concentrated on the distinct features and shortcomings of the SOAP and REST services with respect to the smartphone platforms [14]. That is, the SOAP service uses a standard XML-based message envelope and a set of rules for exchanging structured information which means it has a significant message payload, a parser, and an explicit interface description with WSDL. The REST service, on the other hand, is a lightweight implementation which uses HTTP and URIs to characterize its addressability, uniformity, connectivity and statelessness [14]. Thus, the literature indicates that these inherent characteristics leverage for using complexity, performance, flexibility, footprint, reusability, usability, and scalability as evaluation criteria (see Table 1).

Table 1. Literature summary on the evaluation of Web service technologies

Category	Technology	Criteria						
		Complexity	Performance	Flexibility	Footprint	Reusability	Usability	Scalability
Types of Web services	SOAP services	Generating client code from WSDL requires tools and resulting in **high** complexity [13]	The tools used to consume SOAP degrade performance to **low** [14]	Client code is **stiff** to change because of the structured nature of WSDL [9]	SOAP envelope for operations attributed to **more** footprint [9]	Its highly structured nature **restricts** reusability [23]	The lengthy process required to locate, contact, and invoke makes it **less user friendly**	Tightly coupled and **less** scalable [28]
	REST services	Light weight design makes **less** complex even on thin clients [25]	Significantly **faster**	REST is designed for loose coupling [25], hence **flexible**	Directly invoked with HTTP, **no overhead** [26]	Virtually **unlimited**	Simple but the lack of tools attributed to more **user concern**	Loosely coupled and the **Web** is evident for its scalability
Messaging formats	XML	Text based formats are more easier for human readers but parsing is **complex** for machines	Quantitative results in [16] showed XML's serialization speed is the **slowest**	Human-readability of text-based formats makes them **better** choice for flexibility [16]	Results in [16] showed it produces the **largest** footprint	Text based data can be reformatted for cross-platform **reuse**. XML and JSON can also be parsed by many APIs	XML and JSON are **better** for readable, large user base, and documentation [16]	Scalability can **be achieved** based on the schemas defined by each format and the context of use
	JSON		Much **faster than XML**		Smaller **than XML**			
	Thrift	Binary formats, can only be parsed by tools makes its usage on smartphones **complex** [16]	**Faster than JSON**	Binary formats are platform specific and **less** flexible	**Smaller** than both text-based formats	Platform dependency makes binary less reusable	Usability is **limited** to the specific platform where the parser is deployed	
	Protobuf		The **fastest** of all		The **smallest** of all formats included in the study			
Composition techniques	Static	**Complex**	The task of composing services is generally is user-assisted which lead to **better** performance	**Stiff**	No article discussed about this, but size of the footprint can be influenced by the language it uses	User involvement **enhanced** the reuse [20]	Static and semi-automatic showed usability **limitation** [4]	**Custom** made
	Semi-automatic	Challenging for non-programmers [20]		study in [4] showed that it is the most **flexible**				**High** end user developer [24]
	Automatic	**Simple**	Transforming natural language **slows** it	**Stiff**		Reuse is **limited**	**Better** user experience	**Limited** [4]
Composition languages	Web 2.0 technologies	**Simpler** than WS-PBEL [28]	**Faster**	Literature showed that it is more **flexible** in use than WS-BPEL	It is described for its **small** footprint size [22]	**High** because of the Web 2.0 technologies [19]	**Less** due to lack of tool support for automatic compo.	Composing REST with Web 2.0 is **more** flexible
	WS-BPEL	**Complex** to track states and transitions	Associated tools **slowed** its performance	It is generally **stiff** [30]	**High** due to verboseness of XML	**Low** due to WSDL's structure	**Better** supports automatic approach	**Lower**
Mobile client development approaches	Cross-platform client	**Small effort** needed [32]	Thick clients have **better** communication speed [14]	**Flexible** for multiple platforms	Thick clients owe **high storage** footprint	Generally, this can be achieved depending on the design goal of the applications	**Less** than native [21]	**Better** due to the absence of centralized host
	Native client	**Complex** to deploy		Platform **dependent**			**Excellent**	
	Web client	Generally **simple**	**Offloads** client-side processing	**Flexible** for updates [21]	**High** communication footprint		**Less** than native	**Lower**

Messaging Format. Messaging is a fundamental function in distributed environment that requires data serialization and de-serialization. Thus, the literature shows performance, footprint, and usability as major evaluation criteria for XML, JSON, Protobuf, and Thrift with respect to their appropriateness for smartphone platforms [16]. In addition, we extend these criteria by adding complexity, flexibility, reusability and scalability for their influence during service composition and invocation (see Table 1).

Composition Techniques and Languages. Here, our literature survey was focused on identifying the criteria for static, semi-automatic, and automatic composition techniques as in [19, 20]; and the WS-BPEL and Web 2.0 languages (see Sect. 2). Thus, we found out that performance, footprint and usability are significant criteria for the resource-constrained smartphone [8, 19]. In addition, the use of WS-BPEL standard with SOAP services versus Web 2.0 with REST have strong influence on performance, footprint, and usability [18, 22]. Hence, we adopted these as our criteria along with complexity, flexibility, reusability and scalability evaluate the technologies with respect to the smartphone platform (see Table 1).

Mobile Client Application. In a similar context, we identified criteria to evaluate the mobile client application development approaches. In this regard, Dalmasso et al. [21] used criteria like software quality, usability, development cost, time to market, adaptability, and extensibility when comparing the native, Web, and cross-platform mobile client development approaches. These in many ways represent our complexity, performance, flexibility, footprint, reusability, usability, and scalability criteria described above with respect to evaluating the client development approaches (see Table 1).

In general, the mapping of the Web service and associated technologies into the smartphone environment is influenced by complexity, performance, flexibility, footprint, reusability, usability, and scalability. Thus, evaluation of the four categories technologies conducted by mapping against the criteria as presented next.

3.2 Evaluating Types of Web Service

SOAP service consumers use the SOAP protocol to locate, contact, and invoke the service while the standard HTTP verbs and URIs are sufficient for the REST [23]. In order for a client to use SOAP Web services, it must generate a client code from the WSDL interface which is quite a complex effort [13]. This complexity is also magnified by the fact that developers must produce the same mobile application for several platforms. REST, however, is designed to operate with thin clients without a strict prerequisite for explicit interface description [25]. Thus, REST is less complex than its SOAP counterpart.

In the SOAP service, the process of generating client code from WSDL must make use of heavyweight parsers which are too heavy for the resource-constrained smartphone [9]. This leads to greater performance degradation of the application that employs SOAP services [14] as compared to the WSDL-less REST counterpart [26].

The SOAP service is highly structured which implies that changing a service requires a significant change in the corresponding WSDL and hence change of client code [26].

Moreover, disseminating such change for multiple smartphone platforms is cumbersome. Although similar descriptive languages like WADL and WSDL 2.0 are proposed for the REST services, they do not represent its resource-centric, self-descriptive, and loosely coupled nature [13, 18]. Thus, making a change in a REST service does not significantly affect client code, which makes it a more flexible approach than SOAP.

In the SOAP services, a SOAP protocol specifies an XML-based message envelope and a set of rules for converting platform specific data types into XML representations [14]. With large message payloads, the footprint for enveloping and setting the rules require more memory to be composed on the smartphone. However, the REST service is directly invoked with HTTP and the response is a representation of the resource itself [25]. Thus, no XML overhead is needed for encapsulating communication interfaces, and specifying input/output data types. As a result, the amount of storage and communication footprint of REST is significantly lower for the smartphone than its SOAP counterpart [26].

SOAP service uses a complex data structure which generally restricts its reusability [9, 23]. Reusability of the REST, on the other hand, takes the advantage of the Web 2.0 paradigm which prefers direct access to Web resources [19]. Similarly, the complex data structure attributed to SOAP makes the composition difficult and error prone [9]. In addition, the lengthy process used to locate, contact, and invoke the service limits developer productivity [23]. These characteristics of the SOAP service limits its usability. Unlike SOAP, however, the URI accessibility of REST services attributes to its ease of reuse. Thus, REST services are widely used to build composite applications [27]. However, they also have usability challenges due to the difficulty of encoding a large amount of input data in the resource URIs and the lack of automated tools [26].

For many of the above mentioned reasons, SOAP services also have scalability limitations. The REST services, however, are designed to support caching and parallelization on URIs. For example, GET responses can be cached in proxies and gateways. Moreover, compared to the ad-hoc partitioning of operations behind SOAP interfaces, REST provides a very simple way to support load balancing based on URI partitioning. Hence, together with the possibility of making stateless interactions, the REST service can enable the building of more scalable systems; and the fact that the whole Web is built on REST principles proves its scalability [28].

3.3 Evaluating Messaging Format

XML and JSON are text-based formats which need to be parsed character by character, hence demand heavyweight parsing [16]. However, Protobuf and Thrift are binary formats and use a technique called positional-binding to store a message's name part of the name-value pair in a separate file [16]. In addition, binary formats incur an extra step because they are language dependent and therefore need to be compiled before use. Thus, both text-based and binary formats exhibit a certain level of complexity.

Sumaray and Makki [16] pointed out that XML produces the largest amount of data, followed by JSON, and then Thrift, with Protobuf being the smallest, hence serialization performance is inverted. A related study by Pentland et al. [7] also confirmed that the data transfer speed of JSON is better than XML. Any change on the data serialization

format of the service being invoked may introduce system faults. As long as the substitute data serialization format and the services are compatible with the client's technology, the choice of data format is not a real concern. However, the human-readable characteristic of the text-based protocols is generally a better choice for subsequent use of the data [16].

Because XML and JSON are easily read by human, the resulting data can be reformatted as required and also opens an opportunity for cross-platform compatibility. In addition, XML and JSON are supported by a multitude of programming language APIs. Thus, a Web service that implements XML or JSON as its data format is generally more reusable [16].

The human-readability characteristic of XML and JSON is especially important for reading the context of the data while debugging and making them more usable than the binary formats. Yet, the usability of XML and JSON prevails over the binary formats because they have better documentation and user base [16].

The scalability criteria of data formats can be seen from the perspectives of the data and the application that uses it. Thus, XML is scalable and the scalability of JSON can also be achieved using JSON-schema[4]. However, the ability to support multiple platforms and the need to minimize footprint size as more services are invoked makes the choice of data serialization format a design trade-off. Thus, the literature shows that text-based messaging and data serialization formats appear to be highly important for smartphone.

3.4 Evaluating Composition Technique and Language

Among the composition techniques described in Sect. 2, the semi-automatic is regarded as difficult for nonprogrammers while the automatic technique is the simplest [4, 20]. Similarly, the complexity of a composition language depends on how the states and transitions are managed. Thus, WS-BPEL is more complex than Web 2.0 as it needs to track the states and transitions between the client and each Web service [28].

Automatic service composition, on the other hand, strives to automate user goals extracted from the request and builds the causal link matrix among the inputs and outputs of the services participating in the composition [20]. Thus, transforming natural language into formal requests, the discovery of services and execution sequencing are all made using an automated tool, which significantly degrades the performance of smartphone applications. However, Web 2.0 applications are lightweight and run faster than WS-BPEL based tools which need to execute a graphical workflow designer, process flow logic template, and the BPEL engine on the client [29].

Arguably, questions of flexibility are not appropriate for static techniques. However, the semi-automatic technique has a higher flexibility compared to the automatic [4]. For the languages, the use of HTML5 together with JavaScript APIs to compose a set of loosely-coupled REST services is more flexible when compared with the WS-BPEL standard which is generally used to aggregate the structured WSDL interfaces of SOAP services [30].

[4] http://json-schema.org/.

Regarding communication and storage footprint, the language of choice is more important than the composition technique. Thus, the Ajax feature of Web 2.0 intrinsically provides an asynchronous function in which a partial page refresh would reduce the amount of content drawn into the smartphone resulting in a smaller footprint, while the verbose nature of XML leads to a greater footprint in WS-BPEL [22].

One of the goals of Web services and compositions is to promote service reuse [4]. When considering the service composition techniques and languages, however, it is imperative to explain the reusability of the end result instead of the techniques. The lack of user involvement in the automatic composition limits reusability of the end result, however. Hence, static and semi-automatic techniques provide an opportunity for users to manipulate the result for future reuse [20]. Similarly, reusability of a Web service resulting from WS-BPEL is restricted by the highly structured nature of the WSDL [23]. HTML5/JavaScript APIs, on the other hand, takes advantage of the reusability of Web 2.0 [19].

The automatic composition provides a system for managing services' compatibility [4] which makes the process more user friendly than static and semi-automatic; and the set of automated composition tools associated with WS-BPEL makes it preferable in terms of usability. However, the promise of uniform service interface standards has proven elusive, and absolute automation is unattainable which makes the usability arguable [28].

Web 2.0 is rich in user experience, however, the REST lacks a formal framework to describe, find and orchestrate the services [20, 28]. Thus, the capability of the Web 2.0 technologies is restricted to the use of static or semi-automatic, and the choice of a composition language is a technology selection tradeoff with respect to usability criterion.

Wajid et al. [4] also revealed that the automatic composition technique has limitations in scalability, as it is criticized for its not accommodating user preferred services on templates. The impact of a composition language on scalability is, however, more dependent on the service itself, implying that REST based applications developed with Web 2.0 technologies are more scalable than SOAP service composed with WS-BPEL [28]. In general, our survey indicated that the choice of a composition technique and language generally influences the composition of both the SOAP and REST services.

3.5 Evaluating Mobile Client Development Approach

In the mobile Web client, entering a URL into a general purpose mobile Web browser is the only necessary condition that a user is required to fulfill, which makes it a simple process. The cross-platform and the native mobile clients, however, need to be searched, downloaded, and installed on the mobile client before their actual use. The native approach is even more complex for service consumers because it requires recompiling the resulting code for many different smartphone platforms [21].

The performance of a mobile client application can also be influenced by its processing capacity and communication latency [14]. In fact, the processing capability of smartphone appliances is growing fast and surpasses the bandwidth and latency of a wireless infrastructure where a thick client plays significant role. For example, in the context of developing countries, the quality of installation and maintenance of wireless

data communication infrastructure is much less than the market penetration of state-of-the-art smartphones which are usually limited to offline use. In addition, a mobile client should be thick enough to meet the fast response rate of interactive business applications [6]. Thus, while native and cross-platform approaches can implement a thick client and enhance communication performance by placing a considerable processing and caching load on the client side, the Web client does the converse and offloads client side processing.

Flexibility is another important aspect of a mobile client application that describes the ease and cost of propagating changes to each mobile user. In this case, a Web client (mobile Web browser) just sends requests to the server and presents the response content back onto the screen, which makes it the most flexible approach of all. This is followed by the cross-platform approach for its deployment on many platforms, with the native client being the least flexible [21]. A mobile Web client avoids the need for client side processing and storage, but, also incurs a considerably large amount of communication footprint.

In view of reusability, it is important to mention the fact that mobile clients can employ composition tools to aggregate multiple Web services together and result in specific applications or other reusable services [20]. That is, a mobile Web client owner executes the logic of a composite application which is exclusively located on a remote server and also stores the resulting data back. In the cross-platform and native mobile client approaches, however, the logic of a composite application and its data are stored on the smartphone. Thus, depending on the design goal, all mobile client approaches can be used to generate reusable results.

Usability is another criterion that is employed in the selection of an appropriate mobile client development approach. Thus, studies reported in [21] revealed that the native approach has the highest usability, and that the cross-platform approach provides a user experience similar to the mobile Web.

Scalability is also an important aspect in the design of a mobile client because a growing number of connected mobile clients can lead to the risk of a single point of failure, with consequent limitations in extensibility. Thus, the Web client is highly centralized while the thick client approach (cross-platform and native) would enhance scalability [6, 31]. In general, the choice of a mobile client application development approach generally influences the characteristics of a service composition environment on the smartphone.

4 Results

Our analysis has identified SOAP and REST services as the main candidate technologies for service composition on smartphones. Regarding SOAP, studies in [8, 26] suggested kSOAP2 and kXML2 should be used for resource-limited smartphones. However, Mohamed and Wijesekera [26] pointed out that REST services are generally less resource consuming and more efficient to implement on smartphones. That is, the fact that REST services focus on the description of resources rather than describing how operations are performed presents a fundamental difference for consumers compared to

the use of SOAP [27]. Thus, although SOAP services have been in use for long and appear as the only choice, REST services have come into the frame backed by quantitative data as a better choice for building flexible, scalable and loosely-coupled applications on smartphones (see Table 1).

In the messaging and data serialization formats category, the text- binary feature has made clear distinctions. That is, the text-based formats are generally characterized as human-readable and platform independent, and the quantitative data in [16] reveals their low serialization speed and large footprint compared to binary formats. In addition, Sumaray and Makki [16] recommended that unless there is a compelling reason, XML must be avoided from mobile applications. However, although JSON's serialization speed is a bit lower than that of the binary formats, its human-readability and platform independence outweigh this slight performance limitation.

Our analysis has also revealed important characteristics of the composition techniques and languages in view of implementing on smartphones. Thus, as described in Table 1, the semi-automatic composition technique prevails in respect of many of the specified criteria, with the exception of usability. Similarly, although Web 2.0 technologies lacks standard tool support for composing services, it is best suited for the smartphone when compared to WS-BPEL in respect of many of the criteria, including performance. In addition, the capability of Web 2.0 technologies to write and parse JSON data on any platform leverages its use on smartphones.

The mobile client application development approaches category also showed significant prospect. That is, the literature-based summary in Table 1 demonstrates that among other criteria, communication performance of the native and the cross-platform mobile client approaches outweigh those of the mobile Web client, while being characterized by a lower processing performance and a high storage footprint.

Thus, the cross-platform approach can be tweaked just like the native approach for better usability and reaching a large number of users of different platforms; and it is more appropriate to design a thick cross-platform client to compensate for the communication bandwidth and latency limitations of the smartphone.

Overall, our analysis of the state of the art of technological readiness for Web service composition on smartphones has portrayed important insights into the selection of appropriate technologies. Accordingly, we present our findings based on the features and shortcomings identified during the mapping of technologies to the smartphone platform as follows.

- The REST service works based on limited standards and tools, however, its lightweight, easily accessible, scalable, and self-descriptive design makes it more appropriate for service composition on smartphone.
- Semi-automatic composition of REST services using Web 2.0 technologies enables the achievement of flexibility, performance, and simplicity. In effect, the matching of Web 2.0 technologies with the REST service [22] can reward certain limitations of the recommended technologies such as easing usability during semi-automatic composition.
- Regarding the messaging and data serialization formats, we highlighted earlier that JSON has an acceptable data serialization speed and that it can be written and parsed

on any platform. In addition, the fact that JavaScript contains APIs which can easily parse JSON data makes it well-matched to Web 2.0 technologies provisions.

- Similarly, the cross-platform mobile client application development approach is found more appropriate for reasons explained in the above sections. In addition, our previous research [24, 32] pointed out that a cross-platform approach can be used to implement a REST service composition tool with Web 2.0 technologies.

Overall, the REST services, semi-automatic service composition with Web 2.0 technologies, messaging and data serialization with JSON, and the use of cross-platform mobile client approaches all together constitute a stack of appropriate technologies.

5 Conclusion

The purpose of this paper is to conduct a literature-based survey of available Web service and associated technologies relevant to compose Web services on the smartphone environment. Accordingly, because REST services are light-weight, easily accessible, scalable, and self-descriptive, they are found more appropriate for composing Web services on smartphones. In the same setting, semi-automatic composition with Web 2.0 technologies demonstrated appropriate combination of composition approach and language mainly because of better flexibility, performance, and simplicity.

In the case of messaging and data serialization formats, JSON is found more appropriate given its human-readability, platform independence, acceptable data serialization speed and ease of data parsing with JavaScript APIs. Similarly, approaches for mobile client development are evaluated and the cross-platform approach was found to best suit our study's criteria, mainly for compensating potential bandwidth and latency limitations, a better user experience and reaching a large audience while fully exploiting a smartphone device's capability.

In general, the REST service, semi-automatic composition with Web 2.0 technologies, messaging and data serialization using JSON and the cross-platform mobile client approach constitute a stack of appropriate technologies for the composition of Web services on the constrained smartphone platform. However, these set of technologies revealed usability limitations due to lack of standards and automated composition tools. In addition, issues such as security are not included in the scope of this study. Our future work will address both limitations.

References

1. Grønli, T-.M., Hansen, J., Ghinea, G., Younas, M.: Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS. In: IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), pp. 635–641. IEEE, May 2014
2. Mesfin, G., Grønli, T.-M., Ghinea, G., Younas, M.: Adopting SOA in public service provision. In: Younas, M., Awan, I., Holubova, I. (eds.) MobiWIS 2017. LNCS, vol. 10486, pp. 279–289. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65515-4_23

3. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: an emerging paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, pp. 1–8. Springer, Dordrecht (2006)
4. Wajid, U., Namoune, A., Mehandjiev, N.: A comparison of three service composition approaches for end users. In: AVI, p. 407, May 2010
5. Lin, J., Wong, J., Nichols, J., Cypher, A., Lau, T.A.: End-user programming of mashups with vegemite. In: Proceedings of the 14th International Conference on Intelligent User Interfaces, pp. 97–106. ACM, February 2009
6. Satyanarayanan, M.: Pervasive computing: vision and challenges. Pers. Commun. IEEE **8**(4), 10–17 (2001)
7. Pentland, A.S., Fletcher, R., Hasson, A.: Daknet: rethinking connectivity in developing nations. Computer **37**(1), 78–83 (2004)
8. Srirama, S.N., Jarke, M., Prinz, W.: Mobile Web service provisioning. In: International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications, AICT-ICIW 2006, p. 120. IEEE, February 2006
9. Beaton, J., Jeong, S.Y., Xie, Y., Stylos, J., Myers, B.: Usability challenges for enterprise service-oriented architecture APIs. In: IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, pp. 193–196. IEEE, September 2008
10. Raman, T.V.: Toward 2 W, beyond Web 2.0. Commun. ACM **52**(2), 52–59 (2009)
11. Mesfin, G., Ghinea, G., Grønli, T-.M., Alouneh, S.: REST4Mobile: a framework for enhanced usability of REST services on smartphones. In: Concurrency and Computation: Practice and Experience (2017)
12. Treiber, M., Scherling, C., Dustdar, S.: Applying SOA Principles on Mobile Platforms (2010)
13. Wagh, K., Thool, R.: A comparative study of soap vs rest Web services provisioning techniques for mobile host. J. Inf. Eng. Appl. **2**(5), 12–16 (2012)
14. AlShahwan, F., Moessner, K., Carrez, F.: Evaluation of distributed SOAP and RESTful mobile Web services. Int. J. Adv. Netw. Serv. **3**(3 & 4), 447–461 (2010)
15. Nurseitov, N., Paulson, M., Reynolds, R., Izurieta, C.: Comparison of JSON and XML data interchange formats: a case study. Caine **9**, 157–162 (2009)
16. Sumaray, A., Makki, S.K.: A comparison of data serialization formats for optimal efficiency on a mobile platform. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, p. 48. ACM, February 2012
17. Tamayo, A., Granell, C., Huerta, J.: Using SWE standards for ubiquitous environmental sensing: a performance analysis. Sensors **12**(9), 12026–12051 (2012)
18. Pautasso, C.: RESTful Web service composition with BPEL for REST. Data Knowl. Eng. **68**(9), 851–866 (2009)
19. Beletski, O.: End user mashup programming environments. In: T-111.552 Seminar on Multimedia, April 2008
20. Laga, N., Bertin, E., Crespi, N.: User-centric services and service composition, a survey. In: 32nd Annual IEEE Software Engineering Workshop, SEW 2008, pp. 3–9. IEEE, October 2008
21. Dalmasso, I., Datta, S.K., Bonnet, C., Nikaein, N.: Survey, comparison and evaluation of cross platform mobile application development tools. In: 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 323–328. IEEE, July 2013
22. Marino, E., Spini, F., Paoluzzi, A., Minuti, F., Rosina, M., Bottaro, A.: HTML5 visual composition of REST-like Web services. In: 4th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 49–55. IEEE, May 2013
23. Erl, T.: SOA: Principles of Service Design, vol. 1. Prentice Hall, Upper Saddle River (2008)

24. Mesfin, G., Grønli, T.-M., Midekso, D., Ghinea, G.: Towards end-user development of REST client applications on smartphones. Comput. Stand. Interfaces **44**, 205–219 (2016)
25. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine (2000)
26. Mohamed, K., Wijesekera, D.: Performance analysis of Web services on mobile devices. Procedia Comput. Sci. **10**, 744–751 (2012)
27. Pautasso, C., Zimmermann, O., Leymann, F.: Restful Web services vs. big'Web services: making the right architectural decision. In: Proceedings of the 17th International Conference on World Wide Web, pp. 805–814. ACM, April 2008
28. Lanthaler, M., Gütl, C.: Towards a RESTful service ecosystem. In: 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), pp. 209–214. IEEE, April 2010
29. Schroth, C., Janner, T.: Web 2.0 and SOA: converging concepts enabling the internet of services. IT Prof. **9**(3), 36–41 (2007)
30. Peltz, C.: Web services orchestration and choreography. Computer **10**, 46–52 (2003)
31. Tanenbaum, A.S., Van Steen, M.: Distributed Systems. Prentice-Hall, Upper Saddle River (2007)
32. Mesfin, G., Ghinea, G., Midekso, D., Grønli, T.-M.: Evaluating usability of cross-platform smartphone applications. In: Awan, I., Younas, M., Franch, X., Quer, C. (eds.) MobiWIS 2014. LNCS, vol. 8640, pp. 248–260. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10359-4_20