RESTful Web Services improve the efficiency of data transfer of a whole-farm simulator accessed by Android smartphones

Mauricio Arroqui[*,c,d], Cristian Mateos[a,b], Claudio Machado[c], Alejandro Zunino[a,b]

[a] *ISISTAN Research Institute - UNICEN. Tandil (B7001BBO), Buenos Aires, Argentina. Tel.: +54 (2293) 439682 ext. 35. Fax.: +54 (2293) 439681.*

[b] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).*

[c] *Facultad de Ciencias Veterinarias - UNICEN. Tandil (B7001BBO), Buenos Aires, Argentina. Tel.: +54 (2293) 439850 ext. 223.*

[d] *Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT).*

Abstract

The relentlessly increasing importance and application of Information and Communication Technologies (ICTs) in Agriculture have given birth to a new field called e-Agriculture, which focus on improving agricultural and rural development through a variety of technologies. In this sense, Agricultural Information Systems (AISs) are distributed sources of information that exploit ICTs to make agricultural processes and decision making more efficient. In order to integrate AISs and therefore build added value AISs, Web Service technologies seem to be the right path towards heterogeneous systems integration. However, there is still uncertain which is the best implementation approach to integrate Web Service-enabled AISs and mobile devices, i.e., the remote information accessors by excellence in rural areas. We comparatively explore the outcomes of employing either Simple Object Access Protocol (SOAP) or REpresentational State Transfer (REST) approaches in a Web Service-enabled whole-farm simulator accessed from Android-powered smartphones. Memory usage was 24% lower in SOAP, but even older and lower-end smartphones have enough RAM to avoid detrimental effects on performance. REST-based approaches broadly incur in less byte transferred compared to SOAP, which has huge implications on costs. That is particularly important when the Internet is accessed via GPRS or 3G protocols and pay-per-byte data plans as in most of Latin America rural areas. However, when unlimited data usage became less costly and more available in such areas, SOAP might be preferred due to the higher maturity of both the protocol and the available developer environments.

Key words: Agricultural Information Systems, Web Services, SOAP, REST, Simugan, Mobile devices

[*] Corresponding author
Email address: marroqui@alumnos.exa.unicen.edu.ar (Mauricio Arroqui)

1. Introduction

There is a rapid development of Information and Communication Technologies (ICTs) applied to Agriculture, such as geo-referenced remote-sensing, on-line sensors, and public databases (weather, markets, etc.). This fact poses a continuous and increasing challenge to design and develop new technological strategies to allow farmers accessing new and high quality data for using them as specific information towards better decision making. Within these ICTs, Agricultural Information Systems (AISs) are rich sources of information that are created, maintained and published for the benefit of farmers and agriculturalists (Laliwala et al., 2006). With quite diverse goals such as real-time data monitoring, recommendation and decision-making support and farm simulation, amongst others, there are increasing examples of AISs published on the Web that expose information and data to end users (Murakami et al., 2007; Liang et al., 2010; Ntaliani et al., 2010; Pimenidis and Georgiadis, 2010).

In this arena, there is now an increasing need for systems integration, in the sense that current AISs normally do not operate only as information sources but also as information finders, extractors and integrators, which means that a single AIS can use as input the information produced by another AIS (Laliwala et al., 2006). Service-Oriented Computing (SOC) is a contemporary computing paradigm that supports the development of distributed applications in heterogeneous network environments (Huhns and Singh, 2005). Basically, SOC is often materialized through Web Services technologies (Bichler and Lin, 2006), a set of well-known standards that enable the construction of software components with well-defined interfaces that can be located and called via ubiquitous Web protocols (Curbera et al., 2003; Vaughan-Nichols, 2002).

Most of the existing Web services have been built by employing the Simple Object Access Protocol (SOAP) (Box et al., 2000), a service construction approach by which Web Services are thought as "operations" that can be executed against the system (or an AIS in our case), each composed of input and output parameters. In other words, from the point of view of a client application, any system is viewed as a remote module with one or more operations. More recently, the REpresentational State Transfer (REST) construction approach (Vinoski, 2008) has appeared as a brand new and cost-effective alternative for building Web Services, which is based on exposing a system via "resources" rather than operations. REST promotes and generalizes the principles of the World Wide Web, in the sense that remote systems (sites) offer client applications (browsers) access to resources (pages) by relying on a reduced set of standard actions (GET, POST, PUT, DELETE, etc.). Although there is a dilemma regarding which approach is better, it is known that REST is more intuitive for modeling stateless –as in HTTP– Web Services (Vinoski, 2008) and therefore accessing information-oriented systems.

Moreover, mobile applications have been appointed as a new alternative for economic and social improvement in rural communities (Cranston, 2010). Current mo-

bile devices come in a number of flavors (laptops, smartphones, tablets) and support several connection mechanisms (2G, 3G, GPRS, HDSPA, Wi-Fi). As an example, the Viticulture Service-Oriented Framework (VSOF) system (Cunha et al., 2010) uses mobile devices for decoding certain physical tags that are placed in the field. Then, by pointing a mobile device to a tag, the field location is inferred, and a viticulturalist may download or upload data such as climatic data or pest incidence from/to a central database. Although processing capabilities and battery lifetime of mobile devices have greatly improved over time, also the quantity of available and needed information has increased. Therefore, there is a need for more suitable technologies to use services from resource-constrained devices. In this sense, the REST approach relies on lightweight invocation protocols (Vinoski, 2008), which might be more smartphone-friendly when remotely accessing information-oriented systems such as AISs.

With the prospective of increasing simulator users through mobile applications, in this work we comparatively explore the outcomes of employing either SOAP or REST approaches in a Web Service-enabled whole-farm simulator accessed by smartphones particularly focusing on the Android platform due to its increasing popularity among users (Butler, 2011).

2. Materials and Methods

2.1. Simugan Web Service

Simugan is a web-based whole-farm simulator, oriented to assist the research, teaching and technology transfer of alternative beef cattle production systems (Machado et al., 2010). Any simulation requires a *scenario*, which contains initial values and conditional rules to manage a farm. Scenarios are built with data entered through different user interfaces, which allow the user to create, save, modify, retrieve or delete her/his own scenario(s). Simulation outcomes are sent to the logged user's mail account as a spreadsheet file.

To design and develop Web Services for Simugan, the key information normally requested by users through desktop Web browsers was detected (Fig. 1). This information includes data about users, scenarios, and running simulations (i.e., general data of all of them and partial results of any in particular).

This information was exposed as seven generic services, forming a *Web Service frontier*:

1. Getting a user's scenarios (`getScenarios`).
2. Starting a simulation using a particular scenario (`runSimulation`).
3. Getting information of a particular running simulation (`getSimulationData`).
4. Getting a user's running simulations (`getRunningSimulations`).
5. Stopping a simulation (`deleteStopSimulation`).
6. Getting a particular user information (`getUserInfo`).

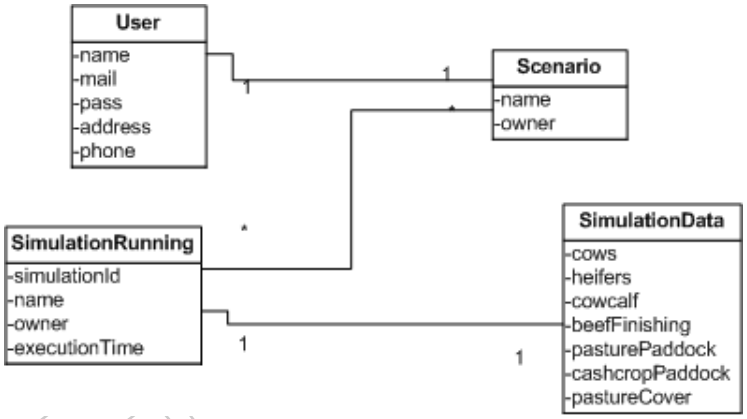


Figure 1: Mapped domain information.

7. Updating a particular user information (`putChangeInfo`).

The technical design and implementation of the above services varied according to the two possible Web Service materializations of the SOC paradigm explored, i.e., SOAP and REST. These are described next.

2.2. The Service Oriented Computing Paradigm

The key concept of the SOC paradigm is the interoperability between different software applications running on a variety of software and hardware platforms (Huhns and Singh, 2005). SOC is known as a loosely coupled architecture (Foster, 2005) and most of its underpinning concepts are developed in the context of Web Services (Singh and Huhns, 2006). The general architectural model for Web Services is composed of a Service Provider (SP), a Service Registry (SR) and a Service Consumer (SC). The SP publishes and unpublishes services to an SR. Then, the SC looks for a desired service in the SR. If the desired service is found, a binding between the SC and SP takes place. This binding could be accomplished by using two conceptually different approaches: SOAP and REST.

Simple Object Access Protocol

SOAP (Box et al., 2000) is a protocol for exchanging information in a decentralized, distributed environment. The protocol specification defines an XML-based (`http://www.w3.org/XML/`) envelope for exchanging messages and a set of encoding rules for converting AIS platform-specific data types into XML representations. This information is contained in a document called Web Service Description Language (WSDL, `http://www.w3.org/TR/wsdl`) that acts as a contract between the SC and the SP for messaging (Curbera et al., 2002).

In the SOAP approach, the seven exposed services were mapped to seven SOAP operations. All these SOAP operations have a request and response message. All

request messages comprise the credentials of the user who starts the request and, in some cases, extra information. For the case of the `runSimulation` operation the scenario identification that will be simulated is needed. In the case of the `getSimulationData` and `deleteStopSimulation` operations the simulation identification associated to the simulation to be returned or stopped, respectively, must be also included. Finally, in case the `putChangeInfo` operation the user mail that will be changed is needed. Response messages are composed by a data type or a primitive integer, long or boolean value that defines the data requested. Thus, our SOAP-based Web Service frontier comprised four data types: User, Scenario, SimulationRunning and SimulationData (Fig. 1).

The implementation of the services was carried out by using the Apache CXF Framework (`http://cxf.apache.org/`) with the contract-first approach to WSDL generation (Erl, 2005; Mateos et al., 2010). We wrote the WSDL and the framework automatically generates Java classes (`http://www.java.com/en/`). In this case, eighteen Java classes were automatically generated, i.e., seven for request messages, seven for response messages and four for the data types. The developer has to maintain the WSDL and an extra Java class that connects the Web service frontier and the business Java model classes.

REpresentational State Transfer

REST (Fielding, 2000) is an architectural approach to SOC which uses basic HTTP methods (PUT, POST, GET, and DELETE) to access a resource applying the correct semantic usage of them. Strictly, a resource is any information that could be referenced by an Uniform Resource Identifier (URI) such as a document, an image or a weather forecast service (e.g., `http://www.weather.gov/forecasts/xml/rest.php`).

In the REST-based Simugan Web Service frontier the GET method was used to retrieve a specific resource, which is analogous to the operations `getScenarios`, `getRunningSimulations`, `getSimulationData` and `getUserInfo` from the SOAP variant of the frontier. The POST method was used to start a new simulation on the server (operation `runSimulation`), which creates a new *SimulationRunning* resource. The DELETE method was used to stop the execution of a running simulation (operation `deleteStopSimulation`), which deletes an instance of the *SimulationRunning* resource. Finally, the PUT method was used to update some user information, such as e-mail, address and phone (operation `putChangeInfo`). In contrast to SOAP, the user credentials and the extra information are added to each URL when accessing each resource. The response messages are composed by representating domain information (Fig. 1) via JavaScript Object Notation (JSON, `http://www.json.org/`).

The Jersey Framework (`http://jersey.java.net/`) and the JSON library were used for implementing the resulting REST resources and encoding data to/from clients, respectively. We implemented the domain information as four Java classes

and one Java class that connects the Web service frontier and the business Java model classes.

2.3. Android Smartphone Experiments

The client side application consuming both the obtained SOAP and REST Web Services was designed and implemented over the Android 2.2 (`http://www.android.com`) platform (Fig. 2). It included four Web Service invocation supports, one for SOAP by using the library ksoap2 (`http://ksoap2.sourceforge.net/`) and three for REST by using Apache's built-in HTTP Connector, the Spring Android Client (`http://www.springsource.org/spring-android`) and the Restlet invocation library (`http://www.restlet.org/`). The three REST libraries present differences about programmability. While Spring provides a complete, easy to use Application Programming Interface (API), in the case of Restlet and Apache, the developer is responsible for parsing the service response. Additionally, in the latter library, the management of Internet connections is also responsibility of the developer. It is worth noting that this client application served only as experimental material to test technological options.

Figure 2: Screenshots of the mobile client developed for calling the implemented Web Services (The implemented application –including its source code– its available at `http://www.exa.unicen.edu.ar/~cmateos/files/android-simugant-client.zip`.).

The evaluation of each invocation support was performed in an experiment that included fifteen runs to get dumps of allocated memory, time elapsed, and bytes

sent/received in the Android smartphone application when calling the operations by using each Web Service implementation client.

3. Results

The average memory usage was lower in SOAP by 24% (Fig. 3). In our experiments, the highest differences regarding RAM allocation were between REST Spring and SOAP (i.e., 0.7 MB). The average time elapsed (Fig. 4a) was similar between each Web Service approach. REST Spring delivered better performance in three services (`runSimulation`, `getSimulationData` and `getRunningSimulations`), SOAP had better performance in other three services (`getScenarios`, `getUserInfo` and `putChangeInfo`) and REST Apache registered the best performance in one service (`deleteStopSimulation`). Regarding average total time results

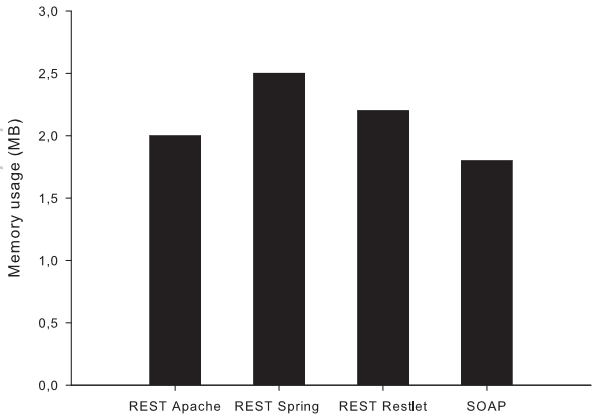


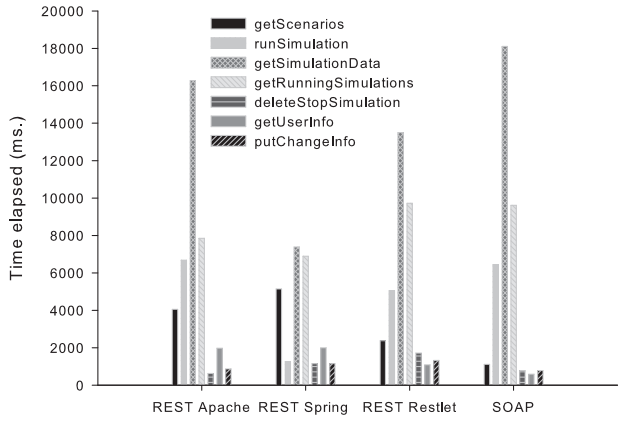
Figure 3: Average memory usage.

(Fig. 4b), REST Spring had better performance than the other Web Service implementations, whereas REST Apache and SOAP delivered the longer times. Additionally, the average between REST approaches was less than SOAP (i.e., 32,701 against 37,432 ms.).

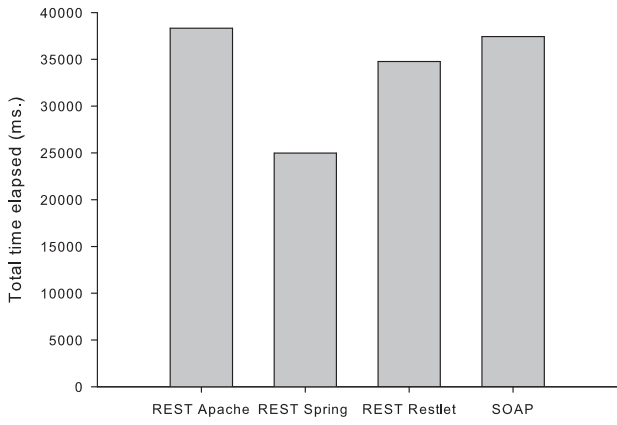
The average bytes sent (Fig. 5a) and average bytes received (Fig. 5b) by the Android application presented a clear difference in favor of the REST-based implementations. In both figures, REST Apache transferred less bytes, followed by REST Restlet, then REST Spring, and lastly SOAP.

4. Discussion

Different authors have compared REST and SOAP from several angles, in order to identify for example how the interactions between distributed parties manifest (zur



(a) Average elapsed time per service invocation.



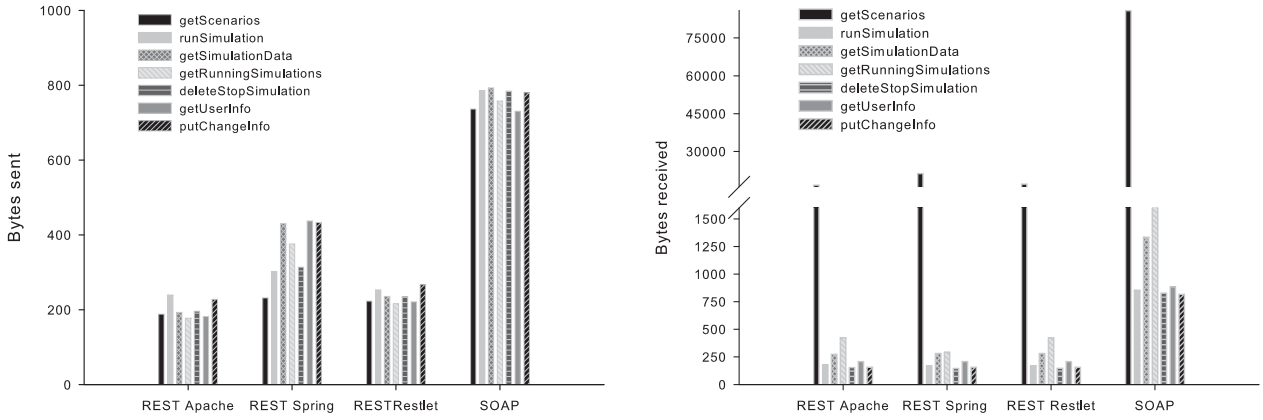
(b) Average total time.

Figure 4: Elapsed time.

Muehlen et al., 2005), how architectural principles and decisions are applied (Pautasso et al., 2008) and how the performance on interacting with stateful resources is (Hamad et al., 2010). To the best of our knowledge, this is the first comparative study consuming stateless (or REST services) and SOAP services of a real AIS from Android-powered smartphones. Smartphones are nowadays the most popular form of mobile device for accessing remote information and the share market of those Android-based ones, has been astonishingly growing (Butler, 2011).

The results showed that the SOAP approach saved 0.7 MB against the worst REST approach in RAM allocation. However, this is not a serious issue, since most smartphones in the market have at least 256 MB of RAM nowadays. Instead, with respect to elapsed time, the REST-based Web Service frontier had a slightly better performance than the SOAP-based frontier. In the case of bytes sent and received the difference was greater, since REST Spring was the worst REST-based approach and was 112% and 310% more efficient than SOAP in bytes sent and in bytes received, respectively.

By the nature of AIS, such as the whole-farm simulator Simugan, many users may need to access it from rural areas. In Latin America, data transference is mostly restricted to 3G and GPRS protocols which present high variability in costs. Therefore, if the Web service will be consumed by smartphones, which protocol will be applied is not a minor issue. For instance, a known international carrier with branch offices in most of Latin America supplies 3G/GPRS access from extreme values of US$ 0.01/MB in Brazil to US$ 1.43/MB in Nicaragua. In the case of Argentina the cost is US$ 0.263/MB for the same carrier, hence our fifteen-round test per Web Service was US$ 0.073 (REST Apache), US$ 0.094 (REST Spring), US$ 0.075 (REST Restlet) and US$ 0.367 (SOAP), which means over costs of 350 % with the



(a) Average bytes sent by android client application.

(b) Average bytes received by android client application.

Figure 5: Bytes transferred.

latter. To illustrate the cost difference by method to a particular user, if a livestock consultant runs in average fifteen simulations by month to model a particular small farm of one of her/his clients, over REST Apache alternative she/he would pay an extra 40, 5 or 530 US$ per year by using REST Spring, REST Restlet or SOAP respectively.

5. Conclusions

Memory usage was 24% lower in SOAP, but even older and lower-end smartphones have enough RAM to avoid detrimental effects on performance. REST-based approaches, particularly REST Apache, provide better performance in terms of bytes transferred versus SOAP, which has huge implications on costs. That is particularly important when Internet is accessed via GPRS or 3G protocols as in most of Latin America rural areas, and data plans are charged per byte transferred. However, when unlimited data usage became less costly and more available in such areas, SOAP might be preferred due to the higher maturity of both the protocol and the available developer environments.

References

Bichler, M., Lin, K.-J., 2006. Service-oriented computing. IEEE Computer 39 (3), 99–101.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D., 2000. Simple Object Access Protocol. `http://www.w3.org/TR/soap/`.

Butler, M., 2011. Android: Changing the mobile landscape. IEEE Pervasive Computing 10, 4–7.

Cranston, P., 2010. The potential of mobile applications for positive social and economic change in rural communities. http://m4agriculture.pbworks.com/f/Mobile+Applications+and+m-Agriculture.pdf.

Cunha, C. R., Peres, E., Morais, R., Oliveira, A. A., Matos, S. G., Fernandes, M. A., Ferreira, P., Reis, M., 2010. The use of mobile devices with multi-tag technologies for an overall contextualized vineyard management. Computers and Electronics in Agriculture 73 (2), 154–164.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S., 2002. Unraveling the Web Services Web: An introduction to SOAP, WSDL, and UDDI. Internet Computing, IEEE 6 (2), 86 –93.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S., 2003. The next step in Web Services. Communications of the ACM 46 (10), 29–34.

Erl, T., 2005. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Fielding, R. T., 2000. Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California.

Foster, I., 2005. Service-oriented science. Science 308 (5723), 814–817.

Hamad, H., Saad, M., Abed, R., 2010. Performance evaluation of RESTful Web Services for mobile devices. International Arab Journal of e-Technology 1, 72–78.

Huhns, M. N., Singh, M. P., 2005. Service-oriented computing: Key concepts and principles. IEEE Internet Computing 9 (1), 75–81.

Laliwala, Z., Sorathia, V., Chaudhary, S., july 2006. Semantic and rule based event-driven services-oriented agricultural recommendation system. In: Proceedings of the 26th IEEE International Conference/Workshops on Distributed Computing Systems. p. 24.

Liang, S., Wenxing, B., Bingbing, L., Guangming, L., Hui, P., 2010. Cow breeding system research based on SOA and Web Services. World Congress on Software Engineering 1, 317–320.

Machado, C., Morris, S., Hodgson, J., Arroqui, M., Mangudo, P., 2010. A Web-based model for simulating whole-farm beef cattle systems. Computers and Electronics in Agriculture 74 (1), 129 – 136.

Mateos, C., Crasso, M., Zunino, A., Campo, M., 2010. Separation of concerns in service-oriented applications based on pervasive design patterns. In: Proceedings of the 2010 ACM Symposium on Applied Computing. SAC '10. ACM, New York, NY, USA, pp. 849–853.

Murakami, E., Saraiva, A. M., Junior, L. C. R., Cugnasca, C. E., Hirakawa, A. R., Correa, P. L., 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Computers and Electronics in Agriculture 58 (1), 37–48, Precision Agriculture in Latin America.

Ntaliani, M., Costopoulou, C., Karetsos, S., Tambouris, E., Tarabanis, K., 2010. Agricultural e-government services: An implementation framework and case study. Computers and Electronics in Agriculture 70 (2), 337–347, special issue on Information and Communication Technologies in Bio and Earth Sciences.

Pautasso, C., Zimmermann, O., Leymann, F., 2008. Restful Web Services vs. "big" Web Services: Making the right architectural decision. In: Proceeding of the 17th international conference on World Wide Web. WWW '08. ACM, New York, NY, USA, pp. 805–814.

Pimenidis, E., Georgiadis, C. K., 2010. Web Services for rural areas–security challenges in development and use. Computers and Electronics in Agriculture 70 (2), 348–354, special issue on Information and Communication Technologies in Bio and Earth Sciences.

Singh, M. P., Huhns, M. N., 2006. Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons, Ltd., Chichester, UK.

Vaughan-Nichols, S., 2002. Web Services: Beyond the hype. Computer 35 (2), 18–21.

Vinoski, S., Sep. 2008. RPC and REST: Dilemma, disruption, and displacement. IEEE Internet Computing 12, 92–95.

zur Muehlen, M., Nickerson, J. V., Swenson, K. D., 2005. Developing Web Services choreography standards–the case of REST vs. SOAP. Decision Support Systems 40 (1), 9 – 29.