ABSTRACT
Although they exist for sometime, service-oriented architectures recently restarted to gain prominence as a technology capable of significantly improvement of the interoperability between software applications. Due to its features of loose coupling and platform independency, SOA is becoming the preferred and more suitable architectural style for ubiquitous or pervasive computing, which also is gaining notoriety and must become an effective reality in a few years. Hence, the need of SOA-based ubiquitous architectures emerged. This paper presents the U-SOA (Ubiquitous Service-Oriented Architecture) and its components that are currently being implemented.

Categories and Subject Descriptors
D.2.11 [Software Engineering]: Software Architectures – domain-specific architectures.

General Terms
Design, Experimentation, Languages

Keywords
framework; web service composition; ubiquitous computing; collaborative environments; impact analysis; SQL; SOA; BPEL; OWL-S; Android

1. INTRODUCTION
Due to the loose coupling and platform independency offered by services-oriented architectures (SOA), which allows the seamless integration of systems built on different platforms, they are being strongly indicated to be the basis of collaborative systems [6]. In the same way, SOA fits very well to mobile systems, including ubiquitous or pervasive computing [8], which is considered the next computer paradigm [12]. SOA is based on three components: service provider, service consumer and services repository. All the traffic between these components uses XML industry standards, like SOAP, for example. The services, which are actually implemented usually as web services, are described with WSDL language. SOA proposes the composition of these services into new composite applications, which conceptually are also web services and consequently are described with WSDL. This composition is done with services composition languages, and the one that is becoming the standard is BPEL [11].

Some improvements in the currently SOA structure that will be available from the proposal architecture are illustrated in Figure 1. The first of them is the possibility of each node of the traditional triangle relationship (provider-repository-consumer) could be any device, including small mobile devices. The second is the utilization of a kind of web services composition description language (WSCDL) for compositions publishing. This new describing way must have needful additional information to identification of the compositions participant services, allowing the mapping of these services and, consequently, several new capabilities, among them the impact analysis quoted in the preceding paragraph. The WSCDL must be defined based on an ontology that represents the framework that is being proposed. This language, which must support the description of services composition as well as of atomic services, will possibly be OWL-S [7], but also it can be created as a WSDL extension or a new composition language, such as SQLSOA (Structure Query Language for Service Oriented Architecture), that is a new composition approach presented in this article.

This paper shows the ubiquitous architecture that is being proposed by our research group, focusing on the tree components that currently are being implemented. The rest of the paper is divided as follows. In section 2, we present the platform on which this work is being implemented. Section 3 explains our architecture. In section 4, related work is presented. Conclusions are argued in section 5.

2. ANDROID PLATFORM
Announced by Google in November 2007, Android [1] is an Open Handset Alliance Project approach for mobile devices platform. The Android platform is a software stack for mobile devices that includes an operational system, middleware and key applications, and it is Linux and Java based. The middleware is composed by a Software Development Kit (SDK) that provides the tools and APIs to develop applications for this platform.

The Android was chosen because it has an extensive library of Java development, free tools and environment, active users’ community, broad support from businesses and because it supplies all the needs and resources demanded by this project.

Considering that Android is developed in Java and that its main libraries were migrated to this platform, nearly all applications written in Java in the market can be easily adapted to this architecture. Thus, kXML [5] and kSOAP [4] libraries can be
used to fill the lack of native support web services for the platform.

![Diagram of Ubiquitous service-oriented architecture.](image1)

**Figure 1.** Ubiquitous service-oriented architecture.

3. UBQUITOUS ARCHITECTURE PROPOSAL

Our ubiquitous architecture (U-SOA) can be seen as technologies stack separated in layers. The layers involved in the architecture contemplate since bottom levels to ubiquitous collaborative applications supported. The architecture bottom is the operational system, for which we have chosen the Android. The layer above the operational system is a virtual machine, to abstract hardware issues. Above this there are the servers or the web services and SOA execution engines. To represent the behavior of these engines, a formal and explicit specification is located in the layer above, provided by an ontology. In top layer there are all the ubiquitous collaborative applications supported, implemented as several services, among them the impact analysis services and the new interface framework based on SQLSOA language. In the next subsections we will focus on three components of U-SOA that are being implemented by now: SQLSOA, SOA engine and glnimpact.

3.1 SQLSOA

SQLSOA (Structured Query Language for Service-Oriented Architecture) is a proposal for a declarative services composition language based on SQL. As shown in Figure 2, this solution involves the specification of a language, a parser, a services orchestrator and a communication layer. The language specification is essential to define the syntax and solution capabilities. The parser will be responsible for lexical analyzes and language semantics. The orchestrator is the component that will coordinate the tree provided by syntactic parser. The communication layer will provide the web services communication.

![Diagram of Components of SQLSOA.](image2)

**Figure 2.** Components of SQLSOA.

The parser component has the language interpretation function making a lexical and semantic analysis of the generated code. The resulting work from this module is a compiled execution tree. This tree contains the sequence of web services requests, the relationship between them and the flow and return details.

The basic structure of a SQLSOA expression to invoke web services consists of three clauses:

- The **select** clause corresponds to an operation of relational algebra projection. It is used to list attributes or desired fields resulting from web services.
- The **from** clause corresponds to a relational algebra Cartesian-product operation. It lists the web services and its endpoints in sequence that should be invoked in the expression assessment.
- The **where** clause corresponds to a relational algebra selection predicate. It consists of a predicate involving the attributes or relations fields that appear in the **from** clause. Its purpose is to carry out the web services operations calls, to restrict and to relate the set of data from the executions.

In the sentence presented in Figure 3 it is brought a set with author and articles written by him/her. The authors' names are caught from a web service of Unisinos papers Authors and the articles from a web service OOPSLA papers. A relationship between articles authors from Unisinos and author articles names from OOPSLA is performed.

```sql
select a.authorName, b.articleName
from http://www.unisinos.br/services/papersAuthors as a,
     http://www.oopsla.org/oopsla2008/services/papers as b
where b.getPapersAuthor.authorName = a.getAuthors.name
```

**Figure 3.** Example of SQLSOA sentence.

The orchestrator has the role of coordinating the syntactic tree provided by the parser. Among the goals of the orchestrator is the function to ensure that the web services are executed in the sequence, in parallel, timing, decision making and also that the data aggregation occurs. This layer is responsible for the work of execution and services composition defined in the SQLSOA sentence. According to [10], an orchestrator must implement the main patterns of workflow to be able to carry out service composition in an effective way. Twenty key patterns features are grouped into six groups that are: basic control flow patterns; advanced branching and synchronization patterns; structural patterns; patterns involving multiple instances; state-based patterns; and cancellation patterns.

The communication layer is responsible for providing the communication services with web services. These services include the WSDL interpretation and SOAP protocol implementation. Both WSDL and SOAP are based on XML, so this layer shall also be able to interpret and manipulate XML. To assist the construction of this layer, kXML and kSOAP libraries will be used. These libraries are already quite widespread and designed specifically to run on mobile devices. kXML will be used to build the XML parser and kSOAP to encode and decode SOAP messages.

The main benefits intended by SQLSOA are:

a) Simplicity and productivity in the statements construction of services composition without the use of graphical tools for the development of scripts, which can be built or edited manually;
b) The ability to be embedded in general purpose languages that need to access services or compose web service;

c) Appropriated to be used on mobile devices with low computational capacity;

d) Flexibility and broad support for major patterns of known workflow.

3.2 SOA Execution Engine

The SOA execution engine for small mobile devices, with the web services execution engine, is responsible for the execution of the functionalities desired for the ubiquitous application, implemented on SOA as web services. The core input of this engine, which also can be called - SOA engine, is illustrated in Figure 4 and explained in larger details as follows.

Component of a major architecture, already explained, the SOA engine communicates with other components by requests. The responsible for receiving these requests is called input and output socket, which listens to determined port until receive some request. The socket is the point of contact of the SOA engine with other architecture components. The socket receives as input parameter the services composition description that must be executed and returns the composition response. These input and output parameters are formatted following some XML standard, for instance. Other possibilities are BPEL or SQLSOA, besides, the XML standard must be ontology based.

Figure 4. Composition execution engine core architecture.

The socket passes the composition description to the central controller, which is responsible to convert it into a cleaner format, excluding the several headers of message used for transporting that occurs by SOAP. Therefore, the central controller uses the incoming handler component. The same way, the central controller is responsible for the correct mounting of the composition response, time that uses the outcoming handler component. The central controller also uses the XML handler to the mounting and unmounting of the input and output parameters of the engine.

The composition manager receives the composition in a clean format, ready for execution. It also uses the XML handler as an assistant component, since in every moment it is XML that is traveling, even internally to the SOA engine's core. The manager composition also utilizes the syntax checker component, which validates the composition description received as SOA engine parameter according to the defined ontology. The composition manager is responsible for the correct execution of each process step. It must be capable of understanding and executing all composition commands, including web services calls, flow control, iterations, etc. The composition manager is the most important component in the SOA engine's core.

To web services executions, the composition manager uses the web services executor component, whose task is to make a call to determined web service and return its response to the component manager. The web services executor can access the services repository to locate determined web service.

The implementation of the SOA engine needs several software components construction. Some of these components, however, can be obtained from existing solutions or even from these solutions customizations. Therefore, we can consider the SOA engine as a collection of technologies to mobile devices, organized and architected with the goal of running web services composition in mobile devices.

3.3 gImpact – An Impact Analysis Framework

The gImpact aims to provide a set of tools to evaluate the impact of changes and the quality of services disposed in mobile devices. This work has the goal of providing a framework that supplies services for impact analysis on services or compositions available on a given server or on all servers that uses the proposed architecture. For example, which would be the impact of the unavailability of S2 service in S1 service composition as presented in Figure 5?

Figure 5. Tree view of services composition.

An important issue when discussing the execution of services is their availability. Due to deal with services available in a network, those can be removed without all the clients being aware of this change, thus occurring an error at the time of their execution, or may still exist the same service on different servers, while for infrastructure reasons one may have a greater availability than other. The term quality of service (QoS), deals with the availability of services, being they related to different areas such as telecommunications, computer networks or other areas. Therefore, the gImpact framework focuses on those two issues.

gImpact provides a range of services that allows the user to make an analysis on the impact of changes in services, from simple changes in a method signature, until the removal of a particular service. For the analysis in compositions already compiled, it is necessary to consult, through the ontology, the services composition tree.
To quantify the impact analysis of this proposal, in addition to the already submitted factors, as the distance where a service is in the execution tree, other factors will be considered such as quality of connection, if the affected services are at the same server or in other servers, since the services can be hosted on mobile devices. Below, it is the proposed formula:

\[ I(S) = \sum_{S \in S} \left( \frac{1}{D(S)} \right) + L \times (1 - Q) \]

Thus, the proposal was distinguished from the studied model by adding factors of quality, besides the services physical location. Some issues of service quality and affected services location were added in the formula. Below the description of each variable:

- **I** corresponds to the impact;
- **S** to services;
- **D** to the distance that the service is in the changed/excluded service tree;
- **L** to the location of the service, should be 0 if it is on the same server;
- **Q** indicates the percentage of quality in connection with the service.

In this approach, the proposal aims to provide services that perform an analysis of availability of a given composition. The service aims to find the following items:

- Identify whether any of the services that compose the function has suffered changes in its signature.
- Identify whether any of the services has been removed from its original server, proposing to use the same service on another server, if that is found.
- Test the quality of the connection with the required services.

4. RELATED WORK

A web services architecture using the Peer-to-Peer is presented by [2] as an interesting infrastructure solution for ad-hoc systems without any infrastructure, being one step more towards the ubiquitous computing. The Mobile Host [9] architecture is an approach for mobile web services provisioning, and it was developed as a web service handler built on the top of a regular web server. The Sliver [3] project is a SOAP and BPEL execution engine for mobile devices. It took the BPEL from large web server. The Sliver [3] project is a SOAP and BPEL execution engine for mobile devices. It took the BPEL from large web server. The Sliver [3] project is a SOAP and BPEL execution engine for mobile devices. It took the BPEL from large web server. The Sliver [3] project is a SOAP and BPEL execution engine for mobile devices. It took the BPEL from large web server. The Sliver [3] project is a SOAP and BPEL execution engine for mobile devices. It took the BPEL from large web server.

5. CONCLUSION

Service-oriented architectures are an interesting approach to support small mobile devices in ubiquitous environment. Therefore, frameworks to support this kind of applications should continue to be built. In this paper we have presented our Ubiquitous Service-Oriented Architecture called U-SOA and its components that are currently being implemented: a new approach to web services composition for mobile devices with a SQL-based language; an engine for services composition execution in ubiquitous environment; and a framework for impact analysis and QoS services.

6. ACKNOWLEDGMENTS

This work is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil.

7. REFERENCES