Mobile Interfaces for Building Control Surveyors

Jacek Chmielewski, Krzysztof Walczak, Wojciech Wiza

Poznan University of Economics, Department of Information Technology,
Mansfelda 4, 60-854, Poznan, Poland
{chmielewski, walczak, wiza}@kti.ue.poznan.pl

Abstract. The problem of integrating heterogeneous back-end platforms used in public administration has been widely addressed in a number of research and development projects. In such a complex and heterogeneous environment, application of the SOA paradigm can be particularly beneficial. However, in some application domains – such as the Building Control Administration – there is an additional requirement: integration of heterogeneous front-end platforms – including access through mobile devices. In this paper, a new method of creating adaptable mobile user interfaces for applications based on SOA services is described. In the adaptation process the displayed content is adjusted, the best way of presenting content is selected and interaction methods are adapted to the capabilities of the particular mobile device. Therefore, it is possible to easily make any service accessible on any mobile device – which is of great importance to the Building Control Administration surveyors that operate out of office, directly in the construction field.

Keywords: SOA, adaptation, mobile devices, mobile user interfaces.

1 Introduction

One of the main advantages of using the Service Oriented Architecture (SOA) paradigm [1] to create software is the ability to seamlessly integrate diverse systems running on heterogeneous hardware and software platforms. This feature is of special importance for efficient deployment of systems associated with complex business processes. An application domain which is characterized by intrinsic complexity of the underlying business processes is the public administration. Public administration requires close cooperation of multiple participants representing different domains, playing different roles and having different areas of interest. Moreover, these participants may come from both the private sector and the public sector, therefore, in general, they have different ways of operation and are subject to various regulations and constraints. In such a complex and heterogeneous environment, application of the SOA paradigm can be particularly beneficial. In many cases, this paradigm may enable creation of new types of applications that could not be built based on traditional approaches or, at least, may significantly reduce the software development costs.

The problem of integrating heterogeneous back-end platforms using the SOA paradigm has been widely addressed in a number of research and development projects [2][3][4]. However, in some application domains – such as the Building

Control Administration for example – there is an additional requirement: integration of heterogeneous front-end platforms. In many cases, work with an application must be performed in several different environments – on a desktop computer in the administration office, on a notebook computer in the construction office, and on a mobile device in the construction field. These devices usually differ significantly in their communication capabilities and processing power, as well as the offered presentation and interaction methods [5]. Access by mobile devices are especially important and – at the same time – the most difficult to integrate.

Apart from addressing different device capabilities, efficiency of work with an application can be significantly improved by allowing users to adjust the application interface and choose the content and the functions, which they need most in given circumstances. In many cases, the method of presentation of the content itself is important as well. Also, users may differ in their privileges to access different types of content and a device may be used in specific context, including geographical position, time of a day, lighting conditions, etc. All these elements should be taken into account in the development of user interfaces for SOA services.

In this paper, we describe a new method of building adaptable user interfaces for SOA services, called *ASIS – Adaptable SOA Interface System*, and provide examples how this method can be used to build SOA based applications for the Building Control Administration. The system is called *BCSS – Building Control Support System* [6].

By the use of the ASIS method it is possible to provide end-users with a convenient and flexible way of accessing available SOA services. The way of accessing the services can be adjusted to the capabilities of the mobile devices used, to users' requirements, and to the current context (place, time, previous interactions). The adjustment is performed automatically, without explicit actions of the user. At the same time, the method is generic and is not bound to any specific set of services or the underlying business logic.

This paper is organized as follows. Section 2 contains an overview of the state of the art in user interface adaptation. In Section 3, an overview of the ASIS system architecture is provided. In Section 4, the ASIS Service Interface Generator is described. Section 5 contains description of the SOIL language used for building adaptable user interfaces. In Section 6, examples of service visualization templates programmed in SOIL are presented. Section 7 concludes the chapter.

2 User interface adaptation

The problem of dynamic creation and adaptation of user interfaces has been widely studied. Existing solutions can be broadly divided into four categories. The first category are declarative user interface languages, such as MXML [7], XUL [8] and XAML [9]. Interfaces built with these languages can be highly sophisticated, but at the same time interface descriptions are usually complex and require complicated interpretation engines running on the client side. While this poses no significant difficulties on stationary PC platforms, practical implementations on mobile devices such as PDAs and smartphones are highly problematic due to the diversity of these platforms, low processing power available, and limitations of the communication channel. Therefore, most of the currently built mobile interfaces are based on HTML [10]. The second category of solutions are HTML page generators such as PHP [11], JSP [12] and ColdFusion [13]. These tools can be used efficiently for generation of web pages and in fact are used in many practical applications on the web. However, they are not really well suited for building mobile interfaces for SOA applications as discussed in this paper. Missing features include possibility of automatically integrating fragments of interface description coming from distributed application services, high-level application specific language elements, and XML [14] compliance for easy processing and generation of interface descriptions. Third category of systems simply use different versions of the interfaces for accessing with stationary and mobile devices (e.g. goole.com and google.com/m). This approach is very limited in its applicability, leads to inconsistencies and increases the amount of configuration that needs to be performed by end-users of these systems. Finally, in the fourth category of solutions, HTML interfaces are equipped with dynamic scripting features that enable their adaptation to the device capabilities on the client side. This solution, however, does not take into account full context information during the interface adaptation, significantly decreases bandwidth efficiency, and leads to security problems. Additionally, support for dynamic HTML features on mobile devices varies significantly.

3 ASIS Architecture

The *Adaptable SOA Interface System* (ASIS) is responsible for generating user interfaces for SOA applications adapted to the particular user, the currently used access device and the current context. The ASIS framework creates the interface description content on request received from the end-user device, based on interface templates coded in a specially designed language, called SOIL – *Service-Oriented Interface Language*, parameters of the end-user device – either sent by the device or stored in a local database, and the current context of interface generation. The interface description is then sent to the client device and rendered. At the client device a typical web browser or a specially designed application extending context information can be used. The overall architecture of the ASIS framework is presented in Fig. 1.

The ASIS framework consists of four main elements: SOA Interface Generator Module (SIG), SOIL Interface Templates, Content Access and Adaptation Module (ADAM), and Content Upload and Adaptation Module (UMOD).

The *SOA Interface Generator* (SIG) consists of an interface generation server and a client application. SIG processes interface templates encoded in the SOIL language, generates the interface content and sends the final content to the client devices. The SIG is further described in Section 4.

The *SOIL Interface Templates* enable SOA services to be presented to end-users in a user-friendly way. The templates are encoded in the SOIL language further presented in Section 5. SOIL is based on XML. It provides commands that can control the process of interface generation, communicate between themselves, and execute calls to SOA services. SOIL is independent of the content description language used (e.g. HTML, XML or textual PDF [15]). Examples of SOIL Interface Templates are presented in Section 6.

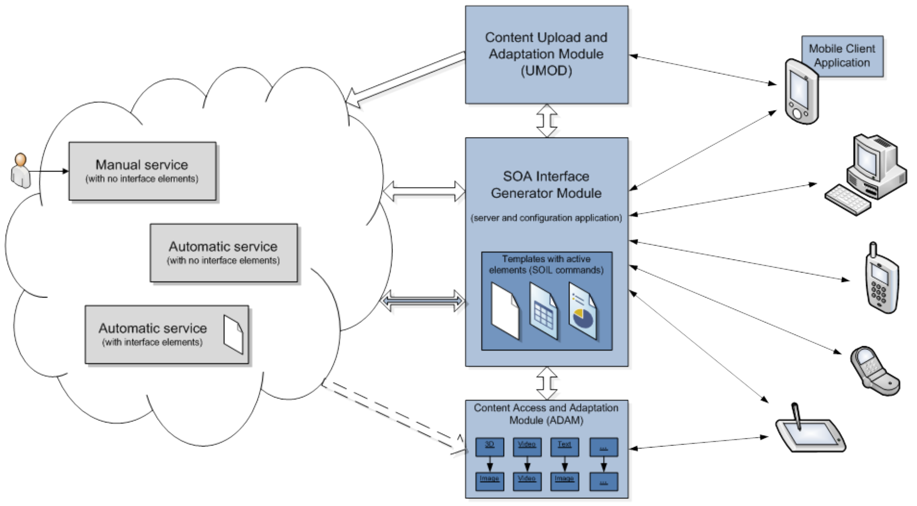

Content Upload and Adaptation Module (UMOD)
Mobile Client Application
Manual service
(with no interface elements)
SOA Interface Generator Module
(server and configuration application)
Automatic service
(with no interface elements)
Templates with active elements (SOIL commands)
Automatic service
(with interface elements)
Content Access and Adaptation Module (ADAM)
3D
Video
Text
...
Image
Video
Image
...

Fig. 1 The overall architecture of the ASIS framework

The *Content Access and Adaptation Module* (ADAM) is responsible for providing access to multimedia objects and adjusting the formats and properties of these objects (type, format, resolution, sampling, precision, compression) to make them suitable for presentation on a particular stationary or mobile device. For example, to display a three-dimensional model of an object on a device, which is not 3D capable, a particular view of the object can be rendered to a 2D raster image. Formatted 2D text can be converted into an image to make the presentation independent of the text presentation capabilities of the particular mobile device.

The *Content Upload and Adaptation Module* (UMOD) is responsible for receiving data from client devices and storing the data in the ASIS Repository or sending it to an application service. In some cases, UMOD can also perform data adaptation operations.

4 The Interface Generator

4.1 Interface Generator architecture

The overall architecture of the ASIS Service Interface Generator is presented in Fig. 2. The ASIS Service Interface Generator consists of the *SOIL Processor* and a collection of *SOIL command implementations*, and uses a collection of *SOIL interface templates*. The SOIL Processor is the main unit responsible for generation of the user interface. The unit contains the SOIL engine, which can interpret SOIL commands contained in the interface templates (cf. Section 6). In response to a request received from a mobile client, the unit generates the final form of the interface description based on the collection of available interface templates and calls to the available *SOA interface services*, which provide the SOIL processor with all information needed in the process of generating the user interface. It includes information about the templates to be used, their parameters, mobile device capabilities, users and their preferences and privileges, etc.

The interface templates consist of fragments of interface description interwoven with SOIL commands. Implementations of the SOIL commands are provided as a collection of Java [16] classes independent of the main SOIL processor. An XML file provides mapping between the language elements and their implementations in the Java classes.

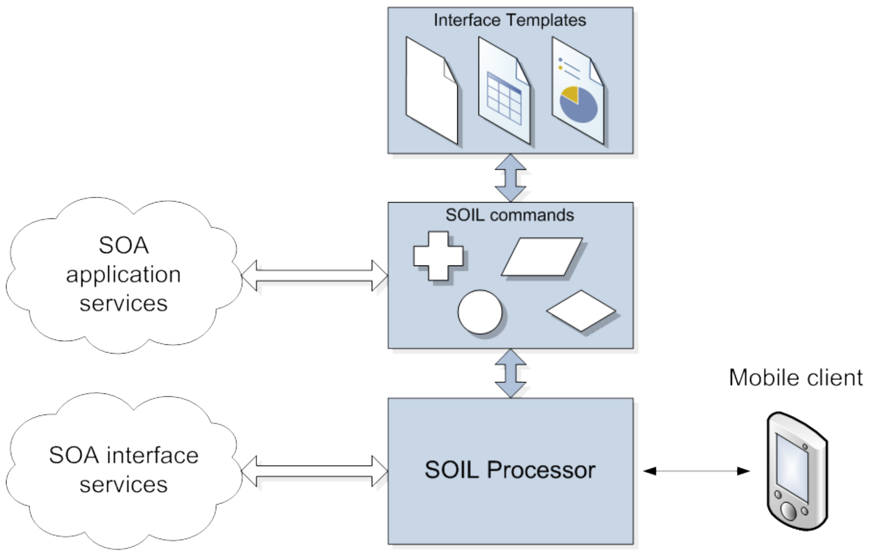


Fig. 2 The overall architecture of the ASIS Service Interface Generator

The SOIL commands can execute *SOA application services*. These are the services, which provide application's business logic. Examples of application services in the BCSS system include retrieval of building information for a given address, retrieval of administrative cases related to the building or storing the list of the selected cases in an electronic 'briefcase' for later access from a mobile device.

4.2 Selection of templates and parameters

To enable differentiation of the content presentation method on different target platforms, ASIS employs a notion of *presentation domains*. A presentation domain corresponds to a target environment, a group of users or a usage scenario for a SOA application. In each application context, a number of templates corresponding to different presentation domains can be used. A template assigned to a particular presentation domain along with some of its parameters defined is called a *template* *instance*. Parameters not defined in the template instance are retrieved from a request sent by the client or the ASIS repository.

To enable automatic selection of template instances based on different and, in general, complex criteria, and to enable automatic setting of template parameter values, a *metamodel processing* phase has been introduced into the interface generation loop (Fig. 3).

With the metamodel approach, the interface content creation process is divided into two phases: *context processing phase* and *interface generation phase*. The context processing phase employs a metamodel, which contains the logic necessary to select appropriate interface template instance and determine values of interface parameters. In the second phase, when the template instance and the values of interface parameters are known, the final interface description is generated based on the selected interface template instance and parameter values.

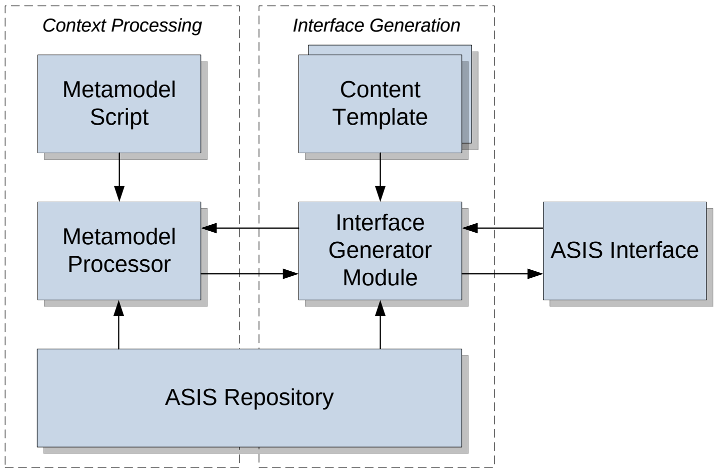


Fig. 3 Metamodel processing in the interface generation loop

5 The SOIL language

SOIL – *Service-Oriented Interface Language* – is a new interface programming language, which has been designed to enable efficient creation of user interfaces for SOA services and applications. The SOIL language enables creation of interface templates combining static elements and dynamically generated elements including results of calls to SOA services.

The SOIL language is based on XML. A SOIL template is an XML representation of an algorithm that generates the interface description. The template is a program built of SOIL commands. The program can execute calls to SOA services, retrieve or update data from/in a database, use values of template parameters, etc. Examples of SOIL commands are: `set` to assign a value to a variable, `for` implementing a numerical loop, `if/then/else` implementing conditional statements, `db_query` to retrieve data from a database, and `soap_call` to execute external SOA services implemented with SOAP. The SOIL commands are encoded in XML and placed inside the text of the interface description written in HTML, XHTML [17] or any other interface description language. The target language can be either XML-based or not. All XML elements that are not known to the SOIL processing unit (e.g., use a different namespace) are ignored and included in the outcome as elements of the interface description.

In SOIL, empty XML elements represent single-line commands such as `<set/>` or `<insert/>`. Non-empty elements represent block-statements like loops, conditions, service calls, database queries, iterations, etc. Examples of non-empty elements are `<for>` ... `</for>` and `<soap_call>` ... `</soap_call>`. The non-empty elements can contain fragments of interface description and SOIL code. The code located inside a repeating element may be interpreted multiple times. Parameters for the command execution are provided in the form of values of element attributes. In SOIL, all parameters are expressions and are evaluated prior to command interpretation.

One of the fundamental concepts in the design of SOIL is extensibility. New language elements can be easily added to the language – either implementing some generic operations or operations specific to a particular application domain. The new elements can take the form of new XML SOIL commands or new functions for use in expressions provided as command parameters.

To facilitate management of the SOIL implementation and documentation, the language has been divided into modules. Specialized SOIL modules with functionality specific for particular domain or application can be added to the implementation. In the BCSS system, four SOIL modules are used:

- the Core Module (SOIL CM), which contains basic language elements independent of a particular application,
- the Database Module (SOIL DB), which contains commands that enable retrieving and updating data in databases,
- the Service Module (SOIL SRV), which contains elements enabling executing requests to SOA services implemented for example as SOAP services, and
- the BCSS Module (SOIL BCSS), which contains elements specific to the BCSS application.

6 Service visualization templates

Service visualization templates have been designed to enable user-friendly bidirectional communication of users with SOA services. Templates are written in the SOIL language (cf. Section 5). The templates can be either generic or application specific and may provide interfaces for single SOA services or for sets of SOA services. Each template provides interface structure and can influence data adaptation by providing parameters for the data adaptation modules (ADAM and UMOD).

A number of SOIL templates have been implemented for the Building Control Support System (BCSS). Different templates have been designed for different devices supporting adaptation of the application interface. In the BCSS environment sample templates have been prepared for the following end-user devices: desktop PCs, netbooks, Tablet PCs, various types of PDAs and smartphones like Toshiba TG01 [18], Apple iPod Touch [19], HTC Touch Pro2 [20] and others. Templates for desktop PCs were developed with standard web-design guidelines and practices. The real challenge was to design templates for mobile devices, where each device can be radically different than the others.

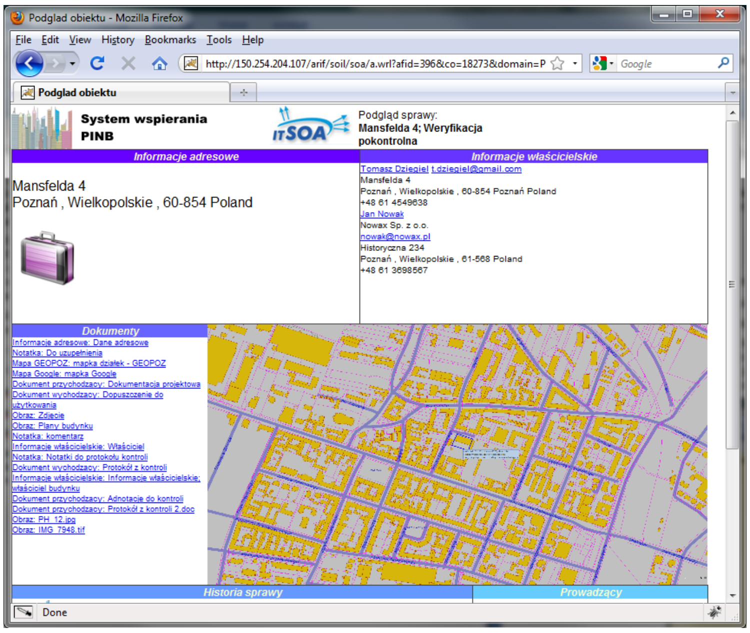
To generate an optimal mobile user interface for a SOA service, the template has to take into account a number of parameters describing device capabilities. This includes a list of possible interaction channels and values of specific parameters describing these interaction channels. The list of available interaction channels limits the forms of communication that can be used by the final user interface. When the form of communication is selected, values of specific parameters are analyzed to decide about information adaptation procedures.

Information about mobile device capabilities and parameters can be stored in a database or provided directly by the mobile device with each request. In the first case, the device type and model, included in the service request, are decoded by the metamodel into one of predefined device groups. Each device group is linked with a specific set of device capabilities and parameters. Such an approach enables to maintain a complete set of information required by a template regardless of the information provided by a device. In the second approach, a special application running on a mobile device extends standard requests with device capabilities and parameters. In some cases, such information is incomplete and cannot be provided directly to the template, but it is still possible to match a specific device group and thus have access to all information required by the template.

The minimum set of parameters required to start ASIS processing is: identifier of the application interface required by the user and the device type and model. These parameters are included in every incoming service request. The identifier of the application interface maps to a set of template instances implementing this interface in different presentation domains. Information about device capabilities and parameters is used to select one of available presentation domains. Combining information about the required application interface and the presentation domain permits selection of the appropriate template instance.

In the case of the BCSS, it was required to prepare templates that allow the user to interact with the content on both desktop PCs and mobile devices. Templates designed for desktop PCs provide access to building/case search and browse functions. For example, the case browsing interface presented in Fig. 4 contains aggregated information retrieved for a particular case:

- Information about the address (police data);
- Owner data (retrieved from court registers);
- List of associated documents;
- Map retrieved from the geodetic office [21] or Google Maps [22];
- Hierarchy of administrative cases related to the building;
- Staff members involved in the administrative case.


Podglad obiektu - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://150.254.204.107/arif/soil/soa/a.wrl?afid=396&co=18273&domain=P
Google
Podglad obiektu
System wspierania PINB
ITSOA
Informacje adresowe
Informacje właścicielskie
Dokumenty
Historia sprawy
Prowadzący
Done

Podgląd sprawy:
Mansfelda 4; Weryfikacja pokontrolna

Mansfelda 4
Poznań , Wielkopolskie , 60-854 Poland

Tomasz Dziegiel t.dziegiel@gmail.com
Mansfelda 4
Poznań , Wielkopolskie , 60-854 Poznań Poland
+48 61 4549638
Jan Nowak
Nowax Sp. z o.o.
nowak@nowax.pl
Historyczna 234
Poznań , Wielkopolskie , 61-568 Poland
+48 61 3698567

Informacje adresowe: Dane adresowe
Notatka: Do uzupełnienia
Mapa GEOPOZ: mapka działek - GEOPOZ
Mapa Google: mapka Google
Dokument przychodzacy: Dokumentacja projektowa
Dokument wychodzacy: Dopuszczenie do użytkowania
Obraz: Zdjecie
Obraz: Plany budynku
Notatka: komentarz
Informacje właścicielskie: Właściciel
Notatka: Notatki do protokołu kontroli
Dokument wychodzacy: Protokół z kontroli
Informacje właścicielskie: Informacje właścicielskie; właściciel budynku
Dokument przychodzacy: Adnotacje do kontroli
Dokument przychodzacy: Protokół z kontroli 2.doc
Obraz: PH_12.jpg
Obraz: IMG_7948.tif

Fig. 4 Case browsing interface for desktop PC

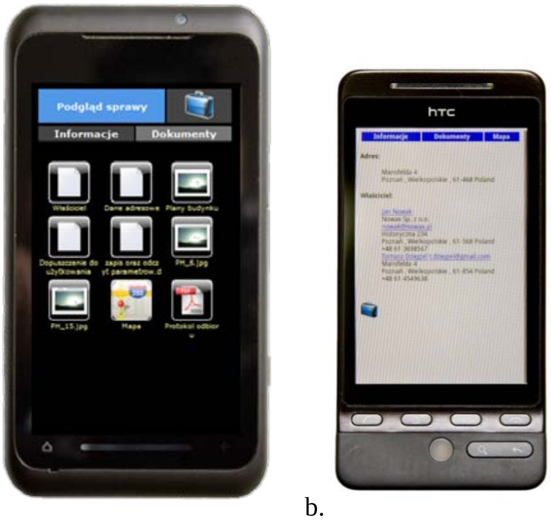
Additionally, each case page contains a briefcase icon. A briefcase is an element permitting a user to select particular case to be included in the mobile interface. By clicking on a briefcase icon the user may select or deselect a given case. By enabling briefcase on selected cases, the user creates a list of cases that will be displayed on his/her mobile device, i.e. that will be accessible outside the office.

When accessing the BCSS system from a mobile device, the user can see a list of cases that have been previously selected by the use of the briefcase icon. The list is displayed in a form adapted to the current device capabilities and size of the display.

A fragment of a SOIL template responsible for generating such a list is presented below.

```
[…]
<SOIL:FLAG method="GETCO" prop="BRIEFCASE" uid="{#uid}" var="colist"/>
<SOIL:SET name="listsize" value="{sizeOf(@colist)}"/>
<span class="hd1">Cases in the briefcase:</span>
<ul>
  <SOIL:FOR from="0" name="i" to="{$listsize-1}">
    <SOIL:CO_PROPS coId="{@colist[$i]}" prop="CO_NAME" var="objName"/>
    <SOIL:EVALUATE>
      <li>
        <a href="...&domain=BCSS.INFO&co={@colist[$i]}">
        <Insert value="{@objName}"/></a>
      </li>
    </SOIL:EVALUATE >
  </SOIL:FOR>
</ul>
[…]
```

If a user wishes to see a case, he/she can click on a particular link. Then the user is presented with the case browsing interface composed of similar sections as for the desktop PC interface. However in this case, the interface is adapted to the current mobile device. For instance, sections can be presented as touch friendly icons (see Fig. 5a) or, on a device with a small display and limited interaction possibilities, entire content can be divided into separate tabs (see Fig. 5b) to make the information easier to access.



a. b.

Fig. 5 Case browsing interface on different mobile devices

7 Conclusions

The ASIS solution for building adaptable mobile user interfaces for SOA applications, presented in this paper, is providing a flexible way of accessing SOA services on mobile devices with limited and non-PC like capabilities. Due to the fact that the ASIS Service Interface Generator generates the final user interface on-demand and takes into account capabilities of the particular mobile device, as well as preferences and privileges of the end-user and additional context information like location, it is possible to provide the user with a final interface in the form most appropriate for a given context, without explicit actions of the user. Adjustment of the final user interface is not bound to any specific set of services or underlying business logic, which enables the ASIS system to be used as a front-end interface generation middleware with virtually any system based on the SOA paradigm. Examples provided in this paper are related to the Building Control Administration, however, the method is generic and can be successfully used also in other application domains.

This work has been partially supported by the Polish Ministry of Science and Higher Education within the European Regional Development Fund, Grant No. POIG.01.03.01-00-008/08.

References

1. Bih, J. 2006. Service oriented architecture (SOA) a new paradigm to implement dynamic e-business solutions. Ubiquity 2006, August (Aug. 2006), 1-1
2. Martins, A., Carrilho, P., Mira da Silva, M., Alves, C.: Using a SOA Paradigm to Integrate with ERP Systems. In: Advances in Information Systems Development, Springer US, 2007
3. Barbir, A., Hobbs, C., Bertino, E., Hirsch, F., Martino, L.: Challenges of testing web services and security in SOA implementations. In: Test and Analysis of Web Services, L. Baresi and E. Di Nitto, Eds. Heidelberg: Springer, 2007, pp. 395–440
4. Karnouskos, S., Baecker, O., de Souza, L. M. S., Spiess, P.: Integration of SOAready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure. In: Proceedings of 12th International Conference on Emerging Technologies and Factory Automation, IEEE Computer Society, 2007
5. Cellary, W., Rykowski, J., Chmielewski, J., Walczak, K., Wiza, W., Wójtowicz, A.: Personalised Access to SOA Services. In: ITSOA project internal report, 2009
6. Building Control Administration – Poznań, PINB – Powiatowy Inspektorat Nadzoru Budowlanego dla Miasta Poznania, http://www.pinb.poznan.pl/
7. MXML, http://opensource.adobe.com/wiki/display/flexsdk/MXML+2009
8. XUL – XML User Interface Language, https://developer.mozilla.org/en/xul
9. XAML – Extensible Application Markup Language, http://msdn.microsoft.com/en-us/library/ms752059.aspx
10. HTML – HyperText Markup Language, http://www.w3.org/TR/html4/
11. PHP: Hypertext Preprocessor, http://php.net/
12. JSP – JavaServer Pages, http://java.sun.com/products/jsp/docs.html
13. ColdFusion, http://www.adobe.com/devnet/coldfusion/
14. XML – Extensible Markup Language, http://www.w3.org/XML/
15. Adobe Mars Project; http://labs.adobe.com/technologies/mars/
16. Java programming language, http://java.sun.com/
17. XHTML – The Extensible HyperText Markup Language, http://www.w3.org/TR/xhtml1/
18. Toshiba TG01, https://www.toshiba-europe.com/mobilerevolution/default.aspx
19. Apple iPod Touch, http://www.apple.com/ipodtouch/
20. HTC Touch Pro2, http://www.htc.com/www/product/touchpro2/overview.html
21. Zarząd Geodezji i Katastru Miejskiego GEOPOZ, http://www.geopoz.pl/
22. Google Maps, http://maps.google.com/support/?hl=en