Design of Service-Oriented Architecture Pattern for Multi-Device and Multi-Platform User Interfaces

Roman Arefev, Tatiana Zudilova
Saint-Petersburg National Research University of Information Technologies, Mechanics & Optics
St.-Petersburg, Russia,
{arefev, zudilova}@ifmo.spb.ru

Ahmed Seffah
Lappeenranta University of Technology
Lappeenranta, Finland
ahmed.seffah@lut.fi

***Abstract*—The paper presents a conceptual Service-Oriented Architecture (SOA) pattern for development of multi-platform and Multi-device User Interfaces (SOA-MUI). This study provides relevant objectives: analyzing the existing patterns for developing SOA-MUI pattern; extending the basic SOA patterns for implementing these interfaces; applying and validating the proposed extended SOA-MUI. The research was done as a step in extending and adapting the principles of SOA while combining the patterns-oriented design and model-driven engineering approaches.**

I. INTRODUCTION

Developing an interactive system and especially User Interfaces (UI) is difficult due to the complexity and the diversity of existing environments and to the amount of skills required. UIs account for more than 50% of the total application costs and development time. In this context, Multiple User Interface (MUI) is among these myriad of innovative UI.

The MUI concept provides multi-views of the same information and coordinates the services available to end-users from different computing platforms [1]. A computing platform is defined as a combination of a hardware device, computing networks, an operating system, and software development toolkits that define the look and feel of the UI. An MUI generally provides support to different types of look and feel. It offers different interaction styles, and takes into account the constraints of each computing platform from which the service is accessible while maintaining cross-platform and multi-devices consistency.

There are many research papers in the field of design application using the SOA approach implementation. However, in these works there is not enough description of implementation issues such as how SOA programming model can be used for the development of human-facing services.

The outcomes of this research[1] are development and validation of a SOA pattern supported by services with a multiple distributed UI that are secure, yet usable and accessible. The pattern is human-centric meaning an explicit incorporation of user experiences into the design and engineering loop.

Developing such interactive systems with a MUI is rather complicated because of the complexity and the diversity of existing environments and to the amount of skills required. The difficulties emerge when the same UI is to be developed for:

- Multiple contexts of use mean different user preferences, categories of users, tasks, locations, and working environments.
- Different devices and operation systems.

The problem addressed in research is the large amount of combinations for possible application implementations on different types of devices with the various operating systems (OS) for example, Windows, Linux, and Android. It should require an innumerous quantity of developers to implement and validate all these possibilities. The high level goal of this work is the attempt of reducing the total application costs and development time as we can not really diminish the possible number of combinations. The novelty is in solution of this problem by the new approach to design of visualizing layer of distributed application, where we applied the new SOA pattern, we proposed for composing the UI services. It allows delivering the correct UI, regardless of OS or type of device.

The paper contains six sections. The next section describes the review of methods and motivation example. Section three includes the analysis of present approaches to implementation of MUI. Section four contains the description of the proposed SOA pattern with explanation of application into the motivating example. In Section five the implementation and used technologies are shown. Discussion and conclusion are in the last Section.

II. BACKGROUND

A. Methodology

We adopted a model-driven approach to modeling with four steps:

1) Analyzing the present approaches to developing multi-platform and multi-devices applications.
2) Specifying architecture models of multi-devices and multi-platform UI pattern.
3) Using SOA design principles for re-architecting multi-platform and multi-devices UI in the form of a set of interrelated services.
4) Applying and validating SOA-MUI model using a

specific application that can be accessed via different multi-platform devices, for instance, interactive tables, tablets, etc.

The design research approach was used as methodology for building and validating the SOA-MUI. A multi-methodological and iterative approach to information system design research includes three actions [2]:

1) *Observation.* We have done it by review of a pool of applications from industry. We sorted out of the practices of developing applications that adapt MUI and inspected the technical specifications of existing frameworks for Human-Computer Interaction (HCI) design. The results can be found in Section three.
2) *Experimentation.* In this stage different design blueprints of SOA-MUI patterns were developed using the Service System Modeling Notation (SSMN) that we developed as a graphical notation (discussed in Section four).
3) *System development.* It consists of five sub-stages: concept design, constructing the architecture of the system, prototyping, product development, and technology transfer. This is a critical activity in the design research process and we developed a SOA pattern for multi-devices UI to support the communication in these five stages. The outcomes are in Sections four and five.

B. Motivating example: "Tires recycling" service

As a case study, we developed a web-service that notifies user (car owners) about the right time to change tires. Every car owner in northern countries needs to change tires for winter seasons. The wear of studs of winter tires can lead to traffic incident. It is important to know the age and condition of the tires, and to make time reservation for the garage visits. It is also necessary to give used tires for recycling to decrease environmental pollution. The demonstrated at Fig. 1 service-oriented system is aimed to assist checking the time for changing the tires and getting the notification about time to change, coordinating the schedule of garage and other places where you can leave the tires for recycling. It also helps to identify involving government and commercial services and regulators.

III. Existing development approaches to MUIs

This section describes the approaches which illustrate the practices of developing applications that adapt to multi-platform devices' constraints and capabilities. The most used and/or promising are presumed the Responsive Web Design (RWD), progressive enhancement based on browser-, device-, or feature-detection, markup language-based approaches, and service-oriented approach to UI (SOA-UI).

A. Responsive Web Design

The main idea of RWD is the adaptation of the layout to the screening environment. It uses fluid, proportion-based grids, flexible images, and CSS3 media queries in the following ways [3]. The fluid grid concept calls for page element sizing to be in relative units like percentages, rather than absolute units like pixels or points. Flexible images are also sized in relative units, so as to prevent them from displaying outside their containing element [4]. Media queries allow the page to use different CSS style rules based on characteristics of the device the site is being displayed on, most commonly the width of the browser. The Flexbox Layout (Flexible Box) module (currently a W3C Candidate Recommendation) is another technology of this approach and it is used for providing a more efficient way to lay out, align and distribute space among items in a container, especially when their size is unknown and/or dynamic.

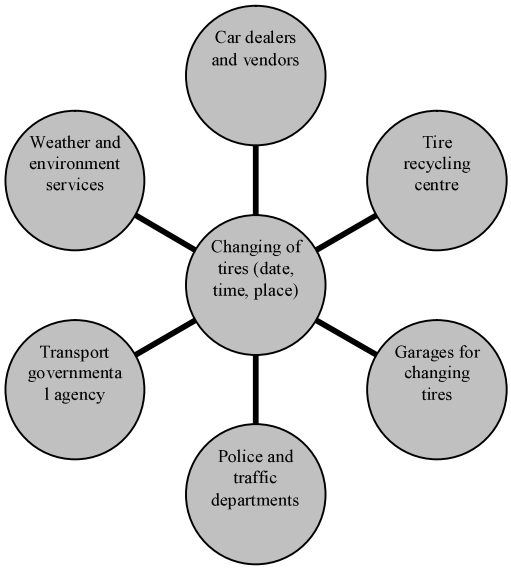

Fig. 1. The main services, which can be accessible via different devices by case study distributed system

B. JavaScript detection

Another way to get UI which can adapt to the device consists of the JavaScript facility. Browser ("user agent") detection (also called "browser sniffing"), and mobile device detection are two ways of determining if certain HTML and CSS features are supported (as a basis for progressive enhancement). However, these methods are not completely reliable unless used in conjunction with a device capabilities database. For a more precise detection of mobile phone and PC features, there are special JavaScript frameworks like Modernizer, ResponseJS, and jQuery Mobile that can directly test browser support for HTML/CSS features.

C. Markup languages

It divides UI markup from application code. In general, User Interface Markup Language (UIML) is a concept in which the data path from the application to the physical display device passes through the abstract field of logic, interface and presentation [5]. The interface area includes a description of the structure, style, content and behavior of the elements. The aim of UIML is to effectively implement the interface area. UIML defines constituent elements of UI, modality of UI elements (visual, verbal, tactile), content which is used in the UI (text, images, sounds, etc.), reaction of the elements of UI to the user, control of events of UI (Java Swing

classes or tags HTML), external Application Programming Interface (API) for interaction with UI [6]. User Interface eXtensible Markup Language (UsiXML) is an another XML-compliant markup language that explains the UI for multiple contexts of application such as Character User Interfaces, Auditory User Interfaces, Graphical User Interfaces, and Multimodal User Interfaces. UsiXML describes UI elements as widgets, containers, controls, buttons, and menus in an abstract way without mentioning the technology it could be implemented with. This makes possible the cross-toolkit development.

D. SOA UI

SOA approach makes easier, faster and less expensive than before to deliver essential information to users. Instead the development of new applications, for instance, when a company enters a new line of business or must tackle with new issues, SOA operates with services (such as the UI or the logic to operate business process) available to users across the network. There is an approach for using SOA not only for providing data but also as provider of visualizing service for this data, which can be obtained from different services. When new business objectives emerge, SOA helps quickly and easily to create new composite applications.

The SOA for UI approach can also be described by Web Services for Remote Portlets (WSRP) specification. It defines a common, well-defined interface to interact with plug-oriented representation of web-services [7]. They are engaged in interactions with users and provide code snippets of UI markup language.

These web-services provide transparent access to content and applications, regardless of their location, and during development suggest a simple visual assembly. This is achieved by unifying the specifications of applications, interface and navigation elements. The WSRP interface descriptions are based on multiple standards - Web Services Description Language and Simple Object Access Protocol. Developed on the basis of WSRP 1.0 web-services can operate on a variety of platforms, including Java/J2EE and Microsoft .NET.

E. Summary

Each of the presented above approaches has advantages and weaknesses. RWD can be implemented with a small amount of code, but it is browser dependent and supporting all possible devices could result in a lot of code. The same is true for progressive enhancement since the devices need to be defined in additional JavaScript files. The advantage of this approach is the almost full control over OS and browser definition. Markup language based approach implies the use of middleware and it cannot be applied all types of devices. SOA approach to UI combines the strengths of markup languages approach with an appliance to SOA principals of loose coupling, autonomy, and composability. It eliminates the weaknesses of RWD and progressive enhancement.

IV. Modeling of SOA based MUI

After the examination of existing approaches to cross-platform UIs we concluded to emphasize on SOA-UI. We are inspired to demonstrate the way to create the distributed system which can produce messages and manage data using a variety of devices and OS. We used already existed technique of visualizing and combine it with developed system design which helped to produce these universal messages.

A. New SOA pattern for monitoring and dynamic design of UI services

The idea of our study is to present the pattern which uses the ability of SOA for composition of existed services for new services. We applied the known SOA patterns as “Service configuration” by Jain and Schmidt [8] and “Capability composition” by Thomas Erl [9] to produce the complete SOA application which has many versions of UI. At Fig. 2 you can see the difference of designed system compared with traditional SOA.

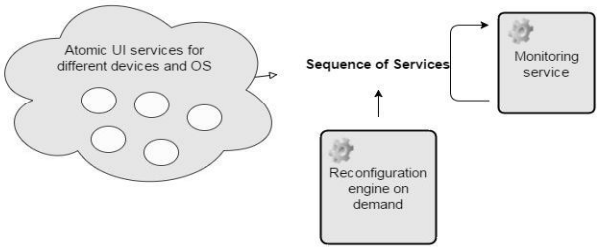

Fig. 2. Pattern for monitoring and dynamic design of UI services

For “Reconfiguration engine on demand” we used the “Service configuration” pattern to switch between different versions of the SOA-UI. The pattern of “Capability composition” aimed to solve a problem that requires logic outside of proposed service boundary.

Fig. 3 shows possible components of a SOA application in relation with our case study detailed in Fig. 1. It depicts the use of the notation we proposed, the service system modeling notation. Basically this notation combines UML (Unified Modeling Notation) and BPMN (Business Process Modeling Notation) to create an advanced notation that helps developers of service systems to specify the structure and behavior of the service systems. It includes the combination of the services or patterns and their relations with the business process. In the presented “tire recycling” application it can be used with different devices, platforms or forms of UI, for instance:

- *Tablet device with Android or Apple iOS*. As a style of interaction, we may use direct touch screen capacity or stylus.
- *Desktop PC*. The UI for this device comprises a keypad, mouse and screen.
- *Mobile phone devices* like Samsung Galaxy or Apple iPhone under the different OS and with different interaction modality such voice, stylus, and touch screen.

As mentioned earlier, one of the long-term outcomes of our research is the SSMN. As a part of this, we propose three

types of plain services based on their functions. The first type is a "Source of data service". The second type is the services for manipulation or transformation of data (business logic service). And the third type is the services for visualizing and interacting with data ("Service for visualizing").

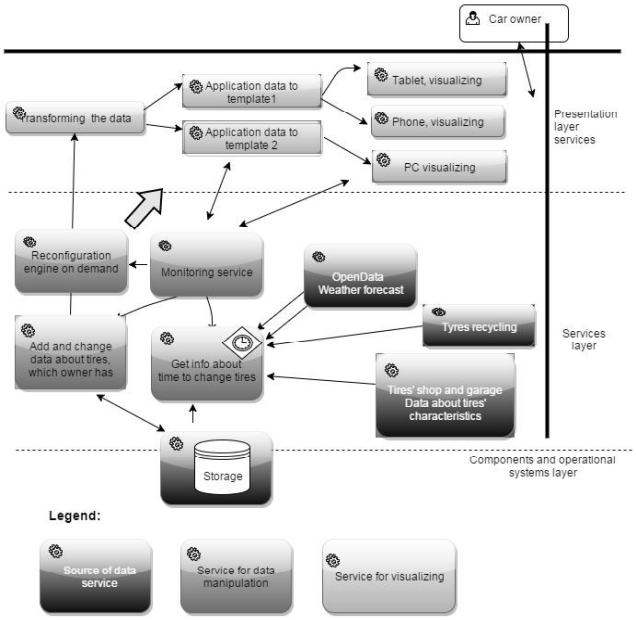

Fig. 3. SOA-based view of distributed application for serving the tires recycling

B. Case study description

The following are the description of main services and message flows that form the proposed case study web-application. The group of "Services for data manipulation" of the proposed application "Tires recycling" according to SSMN is aimed to "Get info about time to change tires in details", manipulating the data of users, monitoring and management of "Services for visualizing" sequence. It collects the info about cars and tires of customers.

The "Services for data manipulation" request "Open Data Weather forecast" source daily about the temperature and checks the calendar (1 December and 1 April) to notify the customer about time when it is better to change tires additionally to checking the age of tires. As an additional option, the "Back-end Service" can request the "Service of Tires' Shop and Garage" to check the characteristics of tires and assist customers to coordinate the right time to change tires. The request to Service of "Tires recyclers" helps to know the place, time and other necessary conditions to leave the used tires for recycling. The request from "Tires recyclers" to the Service of "Tires' shop and garage data" can help to understand the characteristics of tires, the amount of sold tires and their types for balancing marketing and manufacturing.

In our system we had to apply eight principles of SOA [10]:

1) *Services are reusable.* Described services can be reused by any user who provides necessary information. Additionally any other service which provides mentioned information can reuse the tires recycling service.
2) *Services share a formal contract.* The developed service needs exact information to work properly. E-mail to send notifications, current tire type and date of their installation on the car.
3) *Services are loosely coupled.* All services that are used in the presented service are loosely coupled because we do not need to know any details about their implementation.
4) *Services abstract underlying logic.* Tires recycling service hides the underlying logic. There is no need to know how it works to get results. User just has to provide necessary information.
5) *Services are composable.* Our service reuses other services. Among them there are temperature service and service which provides schedule and price.
6) *Services are autonomous.* This service has everything that is needed to calculate the best time for changing tires provided that it has received all necessary information according to the contract.
7) *Services are stateless.* Each query to the service is separate from each other. There is no need to keep track on state information.
8) *Services are discoverable.* The implemented service can be included in all related repositories and can be found easily if necessary.

C. SOMA as a Development Methodology

According to the third stage of design science research approach we have to provide the architecture view of the proposed system. The SOA modeling includes also techniques, which are required for deployment, monitoring, management, and governance to complete SOA life cycle. During the modeling of SOA, it is necessary to decide about architecture of each SOA layer. SOA patterns help to choose the best decision. According to principles of SOA layers' model we have chosen for case study the software development method of Service-Oriented Modeling and Architecture (SOMA) [11]. It helped us to use the holistic approach to design. We follow SOMA stages for the design of our case study application. The first step is service identification, where the processes were described according to business Process Model. Through the following steps of service classification and categorization, subsystem analysis, component specification, and service allocation the SOA model was developed for the case study application. The last step of SOMA is implementation of services. It was made with the help of design patterns for capability composition, service layers, and brokered authentication.

The designed "Tires recycling" service is too large to be

implemented within one single service. It was divided into several: the weather request functionality, recycling production and dealer information, requested from the batch process, part for communication with user and services for monitoring the combinations of services. The UI past affects easily to the mentioned above services according to application logic and User eXperience (UX) without any major changes in other parts of the entire service ecosystem. Employing this pattern can help to achieve the loose-coupling, reusability, and composability principles.

After the decomposition, it was clear that some of our separate services have functional commonality and can be sorted in layers instead of leaving them in one unordered pile which could decrease overall understandability of the service ecosystem architecture. This approach increased the reusability of each service because it was easier to find which service fits better for our needs within preliminary dedicated layers and thus there are fewer chances that in case of ecosystem growth any redundancy will appear.

V. IMPLEMETATION PHASE

A. The development environment

According to chosen research approach we have to prototype SOA application, which shows the implementation of SOA-MUI. In previous section we elicited the architectural principles and now we address the main technologies of implementation.

There are two main parts of our motivation example application in our study which we want to emphasize especially. The first is the group of "Services for data manipulation". These services are not visible to customers and there are no direct UI for them. The "monitoring agent" and "reconfiguration engine on demand" services are included into this group. They manage "The services of visualizing" and produce messages. This notification can be seen by subscribers on their devices using multi-platform environments and UI. It is implemented with appliance of Enterprise Service Bus (ESB) technology using the Integrated Development Environment – Anypoint Studio [12]. It is a well-known technology of rapid development. It allows using the power of Java EE and Spring framework.

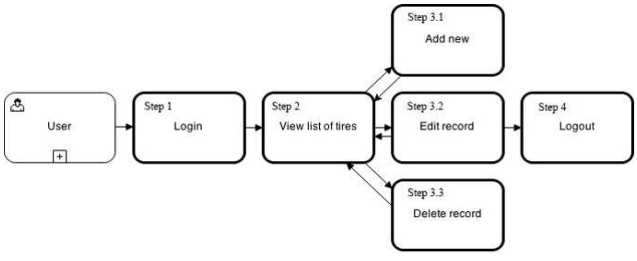

Fig. 4. Sequence data management steps

The second service, which can be detailed, is for managing customers' data regarding the tires in "Storage service". It is based on the Node.js technology for server side and presents API for saving data according to SOA patterns. As a front-end for data input we used rendered HTML page developed according to Model–View–Controller pattern [13] with JavaScript Ajax asynchronous technology [14] with appliance of JQuery library. We use Node.js as a fast and modern technology that simplifies prototyping and has a wide community with a large variety of add-ons. These instruments let to provide the development of scalable and robust applications. One of the additional libraries for brokered authentication via OAuth2 was added to case study application. It uses the authentication service of Facebook according to patterns of HCI. Node.js helped us to implement "Services for visualizing" in a modern way. "Reconfiguration engine on demand" used the approach of React which allowed hot compiling modules depending on request.

B. Data flows and scenarios

The sequence of steps for application data management is presented at Fig. 4. Each of the presented steps implies action with data. It grounds the logic for UI of data management. The data is used in "Back-end service". This service executes the core function of the application. It sets the type of notifications which should be sent based on the incoming information. In our application we made four types of notification and each record should be checked on conditions, which they are connected with. This service is implemented in Java and the Mule ESB technology was used for message flow processing.

Table I identifies scenarios for illustrating the main functions of the application and proves the correct choice of case study topic and necessity of MUI approach in it. Fig. 5 portrays the screenshot of Desktop view and Fig. 6 - Mobile Phone view. Here you can see different views for different types of devices, which are generated according to offered SOA-MUI pattern. Fig. 5 shows the interface for Desktop with large screen and keyboard input, but Fig. 6 depicts interface which was developed for small screen with touch input. This interfaces have been switched during the work of "monitoring agent" and "reconfiguration engine on demand", described in Section four, to change the sequence of visualizing services according to different types of devices with different operating systems (OS).

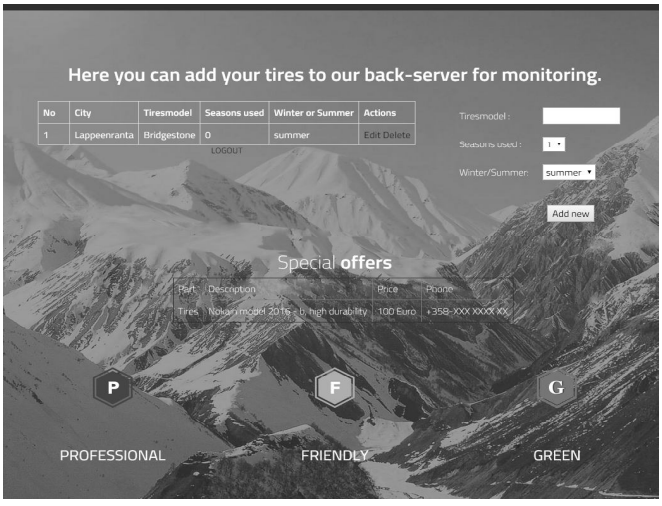

Fig. 5. Desktop view

C. Extension and improvements of research

This research can be aligned with Model-driven architecture (MDA) [15]. MDA suggests transforming one Platform-independent model into several Platform-specific models (PSM), one for each platform or technology where the UI as a service will be operated.

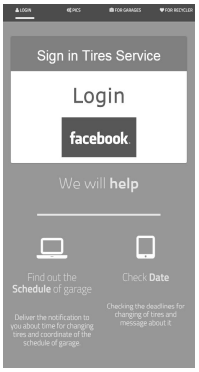

Fig.6. Mobile phone view

MDA also provides support for automatic code generation that implements PSM for those platforms.The SOA-MUI framework paves the direction for modeling UI and user interaction as a part of integrated application development. The following research is able to lead to automatic transformations of UI models into corresponding code. In the future a transition to a NoSQL database should also be studied since it should fit better for SOA and MDA.

D. Related works

We can mention about several existing research works which are aimed to describe the new generation of SOA design patterns.

- Tsai et al. [16] has proposed a technique for dynamic service composition. According to them the SOA composition framework can be distinguished by the following components:

1) *Application Template Registry*. It supplies the application specification with UI requirements pending realization. Developers are able to search, modify, upload and reuse the templates in the registry.
2) *UI Service Registry*. It includes the UIs from the service provider. It provides UI discovery and matching service to the application developer by publishing the UI profile. It also offers application specification template subscription service to the UI services.
3) *UI Composition Service*. It plays an important role in the SOA-UI framework. It allows combining different services to a new one which satisfies developers' needs.

TABLE I. FUNCTIONS AND USED SCENARIOS

Functions	Definition	Example
View tires added to the system	Customer uses the service for management of tires records. After authentication he or she can see all tires.	*Mr. Paterson lives in Lappeenranta. He has family and 2 cars. Totally with summer and winter tires they have 16 tires, but they have also old tires. Mr. Paterson decides to count the tires to understand what tires are necessary to change or send to recycling.*
Add new tires	Customer uses service for managing the tire records and after authentication, he or she can see all tires which system is monitoring, and add new tires with the amount of seasons tires can be used, name of tires, and type (winter or summer).	*Last year you bought a new car and recently you bought winter tires for it. You have used already the Tire Recycling service and want to add new tires.*
Edit options of the tires	Customer uses the service for managing the records about tires and after authentication, he or she can see all tires the system is monitoring, and change the amount of seasons, name of tires, and type of already entered tires.	*Mr. Paterson lives in Lappeenranta. He has family and 2 cars. For summer and winter, the family has 16 tires, but they have also old tires. Mr. Paterson decides to update the property season of records according the current situation.*
Get notification about the date to change tires	The back-end service runs every day and based on the defined conditions, the system decides to send notification in HTML with embedded data via SMTP to user.	*Jessica lives in Helsinki and she has a car. She bought the car in summer and she doesn't know it well yet. Jessica's friend Tony advised her to use Tires recycling, because he has had a car for several years, he knows that used tires should be recycled since it decreases the pollution of environment. Before the 1st of December, Jessica gets notification by e-mail via her tablet including information about the date for changing the summer tires to winter ones. This notification also includes offers of new winter tires from garage and other best offers.*
Get notification about the temperature to change tires	The back-end service runs every day and based on the defined conditions, the system decides to send notification in HTML with embedded data via SMTP to user.	*Jessica lives in Helsinki and she has a car. She bought the car in summer and she doesn't know it well yet. Jessica's friend Tony advised her to use Tires recycling, because he has had a car for ages and he knows that used tires should be recycled since it decreases the pollution of environment. Before the 16th of November, when the weather forecast shows snowstorm and -5 C degrees, Jessica gets a notification by e-mail and opens it at to her tablet. This notification includes information about a date for changing of summer tires to winter tires to decrease the probability of traffic incidents. This notification will also include offers of new winter tires from garage and other best offers.*

1) *Ontology System*. It provides the relationships and searching for UI-related elements. The ontology should include UI classification information.

The weak side of this research is no any points to implementation.

- Gamma [17] has focused on interfaces which can extend the existed functionality of classes, but it is necessary to anticipate the exact methods of interface in design phase and during the improvement of application we need to monitor the dependences.
- Kaminski et al. [18] has talked about combining of different versions and proposed "Chain of adapters" pattern. But this approach means the reconfiguring the whole application, but not combining the separate services.

VI. Conclusion

The standardization of the software development process and the normalization of a UI model are required as never before. SOA appears like an appropriate architectural model. However, SOA is generally discussed in the context of program-to-program interactions. This article describes how the SOA programming model can be used for the development of human-facing services. The review and the case study we conducted suggest that a SOA is suitable for multi-platform and multi-devices UI. The proposed SOA-MUI pattern for provides a baseline pattern for a successful architecture, development and deployment of mobile applications with diverse UI in SOAs.

We first studied four different approaches to the development: RWD, progressive enhancement based on browser-, device-, or feature-detection, markup languages-based approaches, and SOA-UI. We summarized their advantages and weaknesses as a result of this analysis, which showed the perspectives of SOA approach to MUI. The SOA-MUI design process can be aligned with the SOMA methodology. The practical part of this work includes the implementation of the proposed SOA-MUI.

The novelty is a new SOA-UI pattern, used in practice to produce distributed services which can interact with users without limitation of single platform or OS. The SOA-MUI means the new SOA design pattern emphasizing the distributed approach to UI.In a future we see the extension in the research how the proposed SOA-MUI maybe useful for design the successful architecture, development and deployment of mobile applications with diverse UI.

Acknowledgment

The work is supported by the Saint-Petersburg National Research University of Information Technology, Mechanics & Optics (ITMO University) and Lappeenranta University of Technology. We would like to express the deepest appreciation to Professor Uolevi Nikula.

References

[1] A. Seffah, H. Javahery, Multiple User Interfaces: Multiple-Devices, Cross-Platform and Context-Awareness, Wiley and Sons, 2004.

[2] F. Nunamaker, M. Chen, D. M.P. Titus, "Systems Development in Information Systems Research", *Twenty-Third Annual Hawaii International Conference System Science*, vol. 3, 1990, pp. 89-106.

[3] K. De Graeve, HTML5 - Responsive Web Design, Web: https://msdn.microsoft.com/en-us/magazine/hh653584.aspx.

[4] E. Marcotte, Responsive Web Design, Web: https://alistapart.com/article/responsive-web-design.

[5] M. Abrams, J. Helms, User interface markup language (UIML) specification, Web: https://www.oasis-open.org/committees/download.php/5937/uiml-core-3.1-dra%20ft-01-20040311.pdf.

[6] M.F. Ali, M.A. Pérez-Quiñones, E. Shell, M. Abrams, *Building Multi-Platform User Interfaces with UIML*, Springer Netherlands, 2002.

[7] B. Castle, Introduction to web services for remote portlets, Web: http://www.ibm.com/developerworks/ru/library/ws-wsrp/index.html.

[8] P. Jain, P., D. Schmidt, "Service configurator: a pattern for dynamic configuration of services". In Proceedings of the 3rd conference on USENIX Conference on Object-Oriented Technologies (COOTS), vol.3, 1997, pp.16-17.

[9] T. Erl, SOA Design Patterns, 2009. 1st Ed. Prentice Hall PTR.

[10] T. Erl, Service-Oriented Architecture (SOA) Concepts, Technology and Design, 2005, Prentice Hall PTR.

[11] A. Arsanjani, S. Ghosh, S., A. Allam, T. Abdollah, S. Ganapathy, K. Holley, "SOMA: A method for developing service-oriented solutions", *IBM Systems Journal*, 47 (3), 2004, pp. 377-395

[12] Dr. M. Lui, M. Gray, A. Chan, J. Long J., *Pro Spring Integration,* Apress, 2011.

[13] D. Odell, D., *Design Patterns: Architectural,* Apress, 2014.

[14] J.J. Garrett, Ajax: A New Approach to Web Applications, Web: https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf

[15] H.-K.Kim, T.-H Kim, "SOA Modeling Based on MDA", Distributed Computing and Artificial Intelligence, *11th International Conference Advances in Intelligent Systems and Computing*, vol. 29, 2014, pp. 181-194.

[16] W.T. Tsai, Q. Huang, J. Elston, Y. Chen, "Service-Oriented User Interface Modeling and Composition. Research Challenges in Information Science (RCIS)", IEEE International Conference on e-Business Engineering, 2008, pp. 21-28.

[17] E. Gamma, Pattern languages of program design 3, Addison-Wesley Longman Publishing Co, 1997, pp. 79–85.

[18] P. Kaminski, H. Muller, H., M. Litoiu, "A design for adaptive web service evolution", In Proceedings of the International workshop on Self-adaptation and self-managing systems (SEAMS '06), pp. 86–92.