Арефьев Р.А., Зудилова Т.В.

SOA ПАТТЕРН ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ ДЛЯ МУЛЬТИПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ

Аннотация: *В статье представлен новый паттерн проектирования сервис-ориентированной архитектуры (SOA) для мультиплатформенной разработки, применяемый в реализации пользовательских интерфейсов распределенных приложений. В работе ставятся следующие задачи: (1) анализ существующих подходов к разработке мультиплатформенных пользовательских интерфейсов, (2) разработка нового SOA паттерна на основании существующих для применения в разработке мультиплатформенных интерфейсов, (3) тематическое исследование (case study), заключающееся в имплементации предлагаемого SOA паттерна в распределенном приложении и его валидации. В данной работе была использована методология разработки и оптимизации информационных систем, предложенная J. F. Nunamaker. Этот подход итерационен и включает три основных этапа: (1) сбор информации о существующих подходах к архитектуре при разработке Multiple User Interface (MUI), (2) экспериментальная фаза, в которой происходит синтез возможных архитектурных решений, (3) разработка прототипа системы. .В разработанном паттерне проектирования компоновка сервисов визуализации, содержащих различные варианты разметки и вывода данных, происходит внутри одного приложении с помощью механизма динамического мониторинга и реконфигурации в зависимости от характеристик устройства клиента. Может происходить поиск в сети соответствующих сервисов и их динамическое подключение. Практическая значимость результата данной работы – паттерна проектирования SOA для мультиплатформенной разработки заключается в уменьшении стоимости разработки программных продуктов и повышения качества их пользовательского интерфейса.*

Ключевые слова: *облачная информационная система, паттерн проектирования, сервис-ориентированная архитектура, распределенные приложения, пользовательский интерфейс, паттерны SOA, мультиплатформенная разработка, человеко-машинное взаимодействие, адаптивный дизайн, паттерн динамической конфигурации*

Abstract: *The paper presents a new pattern for design service-oriented architecture (SOA) for multiplatform applications applied in the creating user interfaces of distributed applications. The research aims to: (1) analysis of existing approaches to the development of multiplatform user interfaces, (2) development of a new SOA pattern based on existing patterns to be used in the*

DOI: 10.7256/2305-6061.2016.2.18627

development of multiplatform interfaces, (3) case study, which consists of the implementation of the proposed SOA pattern in a distributed application and its validation. In this study, a methodology development and optimization of information systems proposed by J. F. Nunamaker has been used. This approach is iterative and involves three main stages: (1) gathering the information about current approaches to development of architecture of Multiple User Interface (MUI), (2) an experimental phase includes the synthesis possible architectural solutions, (3) development of a system prototype. In the developed pattern design a layout of visualization services, containing different variants of output and markup, is performed within a single application using monitoring and dynamic reconfiguration mechanism according to the characteristics of the client device. A search for relevant services over the internet and their installation is possible. The practical significance of the result of this work is in reducing the cost of software development and improve the quality of their user interface by using new SOA pattern.

Keywords: *dynamic configuration pattern , adaptive design, cloud information system, design pattern, Service-Oriented Architecture, distributed applications, user interface, SOA patterns, multi-platform development, human-machine interaction*

Введение

В связи с распространением применения облачных технологий все большую значимость приобретает проблема универсализации пользовательских интерфейсов для доступа с разных устройств под управлением различных операционных систем (ОС), например, Android в форм-факторе планшетного компьютера и телефона, iOS в форм-факторе планшетного компьютера и телефона, а также стационарных компьютеров под управлением Linux, Windows, OS X. Одна из существующих концепций, обозначенная как Multiple User Interface (MUI) [1] описывает решение в качестве возможности реализации различных представлений посредством комбинации сервисов приложения по визуализации контента в зависимости от типа устройства с которого пришел запрос. Для расчета количества возможных комбинаций пользовательских интерфейсов и отличий в их имплементации возможно применение следующей математической формулы (1):

$$Each_{(D_i,\, DS_j\,,\, Du_b,\, Dm_f,\, Dmod_g,\, Dc_a} \underset{1..n,n,d,c,u,4,t}{\longrightarrow} UIToolkit_{\,l}) =$$

$$\sum\left[\left(D_i \times Ds_j \times Du_b \times Do_c \times Dm_f \times Dmod_g \times Dc_a\right)\right] \times UIToolkit_l \qquad (1)$$

в которой D_i – это различные типы устройств, DS_j – различные типы ОС, Du_b – это различные виды групп пользователей (например, люди с ограниченными возможностями и обычные пользователи), Dm_f: – различные типы контента(мультимедиа, текст, изображения и т.д.), $Dmod_g$, – различия в модальности, Dc_a , – различия в настройках пользовательского интерфейса, также необходимо принять во внимание Do_c – структуру компании, где будет использоваться программное обеспечение (ПО), отдельно стоит от-

DOI: 10.7256/2305-6061.2016.2.18627

метить, что существуют различные типы стилей пользовательского интерфейса, которые обозначены как $UIToolkit_I$. Одним из практических выходов данной работы является возможность уменьшения затрат, связанных с разработкой ПО, так как уменьшение количества возможных комбинаций приводит к снижению качества пользовательского интерфейса.

A. Методология

В данной работе была использована методология разработки и оптимизации информационных систем, предложенная J.F. Nunamaker [2]. Этот подход итерационен и включает три основных этапа:

1. Сбор информации о существующих подходах к архитектуре MUI, их анализ с целью определения наиболее сильных и слабых сторон и перспективности их применения.
2. Экспериментальная фаза, в которой происходит синтез возможных архитектурных решений, на этом этапе была также применен новый графический язык моделирования SOA систем.
3. Разработка прототипа системы. Ведется работа по имплементации предлагаемого SOA паттерна при разработке облачной информационной системы «Утилизации шин». Система позволяет пользователю следить на состояние покрышек своего автомобиля, производить своевременную замену летней и зимней резины, а также найти сервис и продавца, который подберет подходящий тип шин по лучшей цене, а также завод по переработке шин, который может выкупить использованные шины по лучшей цене.

Существующие подходы к разработке MUI

В этом разделе описаны наиболее часто используемые подходы к разработке приложений пользовательского интерфейса (UI) которые могут быть адаптивны к различным устройствам или ОС.

A. Responsive Web Design

Главная идея Responsive Web Design заключается в адаптации разметки UI, представленного в виде какого-либо вида, к экрану, на котором он отображается. Инструментами реализации данного подхода являются изменяемая пропорционально сетка для разметки, зависимый от размера экрана графический контент, CSS3 медиа-запросы [3]. Концепция сетки разметки описывает возможность указания размера элементов UI в зависимости от размера экрана и может указываться как в абсолютных величинах – пикселах, так и в относительных (процент от размера экрана) [4]. Подстраивающийся графический контент позволяет предотвратить перекрывание элементов UI.

Б. JavaScript подходы

Другое направление в разработке MUI представлено применением JavaScript. Суть технологии сводиться к определению типа устройства, ОС, на котором просматривается UI и отображением соответствующей разметки для этого устройства. Недостатки данного подхода – это расширяющееся многообразие пользовательских устройств, ведущее к

DOI: 10.7256/2305-6061.2016.2.18627

постоянному обновлению информации в справочниках Java Script библиотек.

В. Языки разметки UI

Предназначение языков разметки заключается в разделении разметки UI от кода приложения. Наиболее известный язык разметки – User Interface Markup Language (UIML) описывает как данные из приложения будут отображаться на экране устройства с помощью абстракции, содержащей логику отображения и элементы интерфейса [5]. Часть, описывающая элементы интерфейса, содержит информацию об элементах UI, включая информацию о модальности этих элементов (визуальный, вербальный, тактильный), информацию о контенте, который содержат эти элементы (текст, графика, звук, видео), поведении элементов в ответ на действия пользователя, внешних Application Programming Interface (API) для взаимодействия с элементами UI.

Г. SOA для MUI

Подход SOA делает более простым, быстрым и менее затратным поставку информации пользователю. Вместо разработки нового приложения для представления пользователю, например, какого-нибудь нового функционала по проставлению данных, SOA оперирует существующими сервисами для генерации новых. Этот подход позволяет быстро создать композицию нового сервиса, в том числи и для представления визуализации контента.

Одна из возможных реализаций – это Web Services for Remote Portlets (WSRP). Этот подход определяет интерфейс для взаимодействия с веб-сервисами [6], которые вовлечены во взаимодействие с пользователями. Этот интерфейс обеспечивает шаблон кода языка разметки UI. Данные веб-сервисы обеспечивают доступ к контенту и приложениям не зависимо от их расположения.

Д. Выводы

Каждый из описанных выше методов имеет преимущества и недостатки. RWD может быть реализован с помощью сравнительно небольшого объема кода, но он браузерно зависим и не поддерживает всех возможных имплементаций MUI. Указанное верно и для второго описанного подхода с помощью JavaScript, но помимо этого устройство должно поддерживать соответствующую реализацию JavaScript, а также как было указано в секции Б требуется постоянное обновление в справочниках по мере появления новых устройств. Сильной стороной данного подхода можно назвать почти полный контроль за версией ОС и браузера. Третий подход или языки разметки подразумевает использование промежуточного программного слоя и нее пожжет быть использован на устройствах всех типов. SOA подход к UI объединяет сильные стороны подхода языка разметки с применением таким принципов SOA как слабая связанность, автономность, способность к композиции, тем самым убирая недостатки трех первых обсуждавшихся подходов.

Паттерн динамической конфигурации MUI

Основной задачей работы является разработка SOA паттерна для применения в MUI и разработка распределенного приложения, демонстрирующего его возможности. Данная научная статья освещает первую часть задачи – разработку архитектурного решения в

DOI: 10.7256/2305-6061.2016.2.18627

виде паттерна.

А. Принципы SOA паттерна для мониторинга и динамической компоновки сервисов визуализации пользовательского интерфейса

В этом исследовании мы основывались на уже признанных паттернах проектирования SOA таких как «Сервис конфигурации» Jain и Schmidt [7] и «Способность к композиции» Thomas Erl [8]. На Рис. 1 представлен основные отличия предлагаемой композиции от существующих паттернов SOA. Для «Реконфигурационного механизма» был использован паттерн «Сервиса конфигурации» для переключения между различными сервисами визуализации UI. Паттерн «Способность к композиции» призван решить проблему нехватки сервисов визуализации посредством поиска и композиции атомарных UI сервисов. Как видно из рисунка предлагаемый паттерн позволяет итерационно отслеживать последовательность сервисов, включенных в цепочку для передачи данных потребителю, и динамически менять последовательность, подбирая необходимые атомарные сервисы из доступных в репозитории.

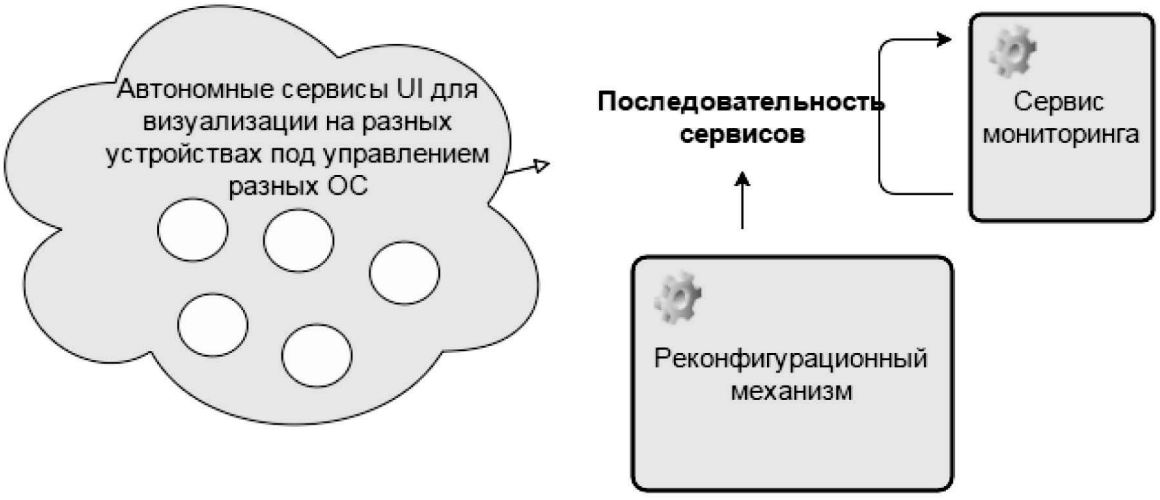

Рис. 1. SOA паттерн для мониторинга и динамической компоновки сервисов визуализации

Б. Пример реализации предложенного паттерна SOA

Для лучшего понимания были выделены три группы сервисов в SOA при проектировании приложений, имеющих UI. Первая группа – это сервисы получения данных. Вторая группа – это сервисы, осуществляющие обработку и передачу данных дальше (эту группу также можно назвать группой бизнес логики). К третьей группе можно отнести сервисы для визуализации и представления данных. Для валидации предложенного паттерна было предложено тематическое исследование (case study) – разработка приложения для помощи владельцам автомобилей в своевременной замене зимней и летней резины, поиске сервиса для замены по лучшей цене, поиске центра по переработке шин, который готов выкупить старые шины для утилизации.

На Рис.2 представлены возможные компоненты такого приложения соглас-

DOI: 10.7256/2305-6061.2016.2.18627

но принципам слоев архитектуры SOA и методологии Service-Oriented Modeling and Architecture (SOMA) [9], предложенной группой исследователей компании IBM.

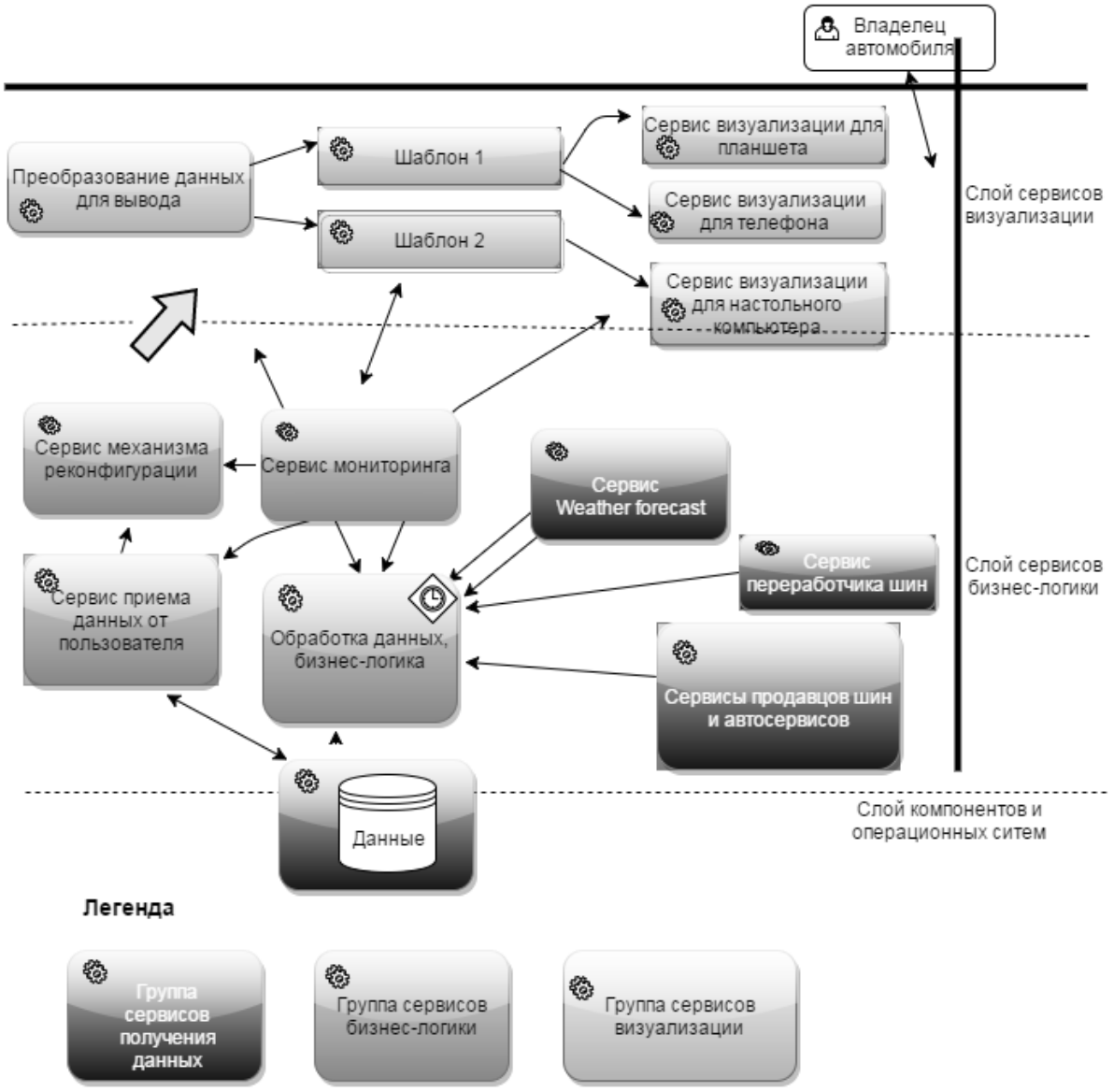

Рис.2. Пример реализации предложенного SOA паттерна

Группа сервисов получения данных поставляет данные от различных сервисов прогноза погоды, продавца шин, сервиса по замене шин, производства по переработке шин и передает их группе сервисов бизнес-логики. В одном из этих сервисов происходит обработка данных пользователей, данных от сервисов данных и формирование сообщения для пользователей. Далее вступает в работу предложенный паттерн мониторинга и динамической компоновки. На основании информации о типе устройства, на котором открывается сообщение, происходит формирование интерфейса отображение сообще-

DOI: 10.7256/2305-6061.2016.2.18627

ния и взаимодействия с пользователем. Данный интерфейс динамически настраивается в зависимости от того, куда передается поток вывода данных.

Обсуждение SOA паттерна для динамической конфигурации сервисов визуализации и результатов других авторов

На основании проведенного литературного поиска было выделено несколько подходов к проектированию динамически изменяющихся приложений:

1. Tsai с соавторами [10] предложил фреймворк по динамической композиции сервисов. Данный фремворк включает следующие компоненты:
   - **Репозиторий для шаблонов приложений**. Он содержит спецификации приложений с набором требований к UI. Разработчики могут искать, модифицировать, загружать и использовать данные шаблоны.
   - **Репозиторий UI сервисов.** Он содержит UI от поставщика сервисов и обеспечивает доступность из сети посредством публикации профиля UI.
   - **Сервис композиции UI.** Этот компонент помогает комбинировать сервисы.
   - **Сервис онтологии.** Он включает информацию о классификации UI, а также взаимодействии UI-элементов.

К недостаткам данной научной работы можно отнести крайнюю абстрактность предлагаемого фреймворка без указаний на возможность практического применения в среде разработке приложений.

2. Научные труды Gamma [11] широко известны в SOA. Им было разработано несколько паттернов проектирования, среди них можно выделить подход, основанный на интерфейсах, способных расширять функциональность классов. Однако данный подход, обладает недостатком необходимости заранее описывать всю возможную функциональность, чтобы потом имплементировать в классах. Конечно, данный недостаток можно уменьшить используя подход внедрения зависимостей (Dependency injections), однако он не обсуждается в данной работе, так как эта технология стала применяться несколько позже и не включена в предлагаемый Gamma паттерн.
3. Kaminski и соавторы [12] обсуждают возможность комбинаций различных версий сервисов и предлагают паттерн «Цепь адаптеров», но данный подход означает реконфигурацию всего приложения целиком, а не комбинацию различных сервисов внутри приложения.

В рассмотренных работах авторы не смогли найти адекватные подходы к решению поставленных задач. Что может служить основой для вывода о новизне предлагаемого SOA паттерна и актуальности рассматриваемой темы исследования. Практическое внедрение данного подхода содержит описание конкретной реализации, использующей CASE инструменты, а также принципы SOA, которые позволяют выстроить распределенное приложение согласно SOMA методологии.

DOI: 10.7256/2305-6061.2016.2.18627

Выводы

Методология SOA обычно обсуждается в контексте взаимодействия компонентов приложения при проектировании бизнес-логики. Данная научная работа посвящена исследованию возможности применения SOA подхода в сервисах взаимодействия человек-машина. Предложенный SOA паттерн проектирования показывает на примере тематического исследования – приложения по замене автошин позволяет сделать вывод о возможности его применении применения для мультиплатформенной разработки.

Во втором разделе был проведен анализ существующих подходов к MUI, выделены сильные и слабые стороны подходов, определены перспективные направления. На основании анализа существующих SOA паттернов и связанных научных работ в этой области был предложен SOA паттерн для мониторинга и динамической компоновки сервисов визуализации пользовательского интерфейса. Выполнена апробация работоспособности данного паттерна на примере рассмотренного приложения. Практическая значимость внедрения SOA паттерна для мультиплатформенной разработки заключается в уменьшении стоимости создания программных продуктов и повышении качества их пользовательского интерфейса.

Библиография :

1. Javahery H. et al. Multiple User Interfaces: Multiple-Devices, Cross-Platform and Context-Awareness, chapter 12 "Migrating User Interfaces between Platforms Using HCI Patterns". – 2003. – 414 с.
2. Nunamaker Jr J. F., Chen M., Purdin T. D. M. Systems development in information systems research //Journal of management information systems. – 1990. – Т. 7. – №. 3. – С. 89-106.
3. Frain B. Responsive web design with HTML5 and CSS3. – Packt Publishing Ltd, 2012. – 324 с.
4. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения – Издательский дом" Питер". – 2011. – 318 с.
5. Ali M. F. et al. Building multi-platform user interfaces with UIML //Computer-Aided Design of User Interfaces III. – Springer Netherlands, 2002. – С. 255-266.
6. Castle B. Introduction to web services for remote portlets //IBM Developerworks. – 2005. p. 24.
7. Jain P., Schmidt D. C. Service Configurator: A Pattern for Dynamic Configuration and Reconfiguration of Communication Services. – 1996. – С. 303-307.
8. Erl T., Patterns S. O. A. D. Prentice Hall PTR //Upper Saddle River, NJ. – 2009. p. 65.
9. Arsanjani A. et al. SOMA: A method for developing service-oriented solutions //IBM systems Journal. – 2008. – Т. 47. – №. 3. – С. 377-396.
10. Tsai W. T. et al. Service-oriented user interface modeling and composition //e-Business Engineering, 2008. ICEBE'08. IEEE International Conference on. – IEEE, 2008. – С. 21-28.
11. Gamma E. Pattern languages of program design 3. – Addison-Wesley Longman Publishing Co. – 1997. – С. 79–85.

DOI: 10.7256/2305-6061.2016.2.18627

12. Kaminski P., Müller H., Litoiu M. A design for adaptive web service evolution //Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems. – ACM, 2006. – С. 86-92.

References:

1. Javahery H. et al. Multiple User Interfaces: Multiple-Devices, Cross-Platform and Context-Awareness, chapter 12 "Migrating User Interfaces between Platforms Using HCI Patterns". – 2003. – 414 c.
2. Nunamaker Jr J. F., Chen M., Purdin T. D. M. Systems development in information systems research //Journal of management information systems. – 1990. – Т. 7. – №. 3. – S. 89-106.
3. Frain B. Responsive web design with HTML5 and CSS3. – Packt Publishing Ltd, 2012. – 324 c.
4. Khogan B. HTML5 i CSS3. Veb-razrabotka po standartam novogo pokoleniya – Izdatel'skii dom" Piter". – 2011. – 318 c.
5. Ali M. F. et al. Building multi-platform user interfaces with UIML //Computer-Aided Design of User Interfaces III. – Springer Netherlands, 2002. – S. 255-266.
6. Castle B. Introduction to web services for remote portlets //IBM Developerworks. – 2005. p. 24.
7. Jain P., Schmidt D. C. Service Configurator: A Pattern for Dynamic Configuration and Reconfiguration of Communication Services. – 1996. – S. 303-307.
8. Erl T., Patterns S. O. A. D. Prentice Hall PTR //Upper Saddle River, NJ. – 2009. p. 65.
9. Arsanjani A. et al. SOMA: A method for developing service-oriented solutions //IBM systems Journal. – 2008. – Т. 47. – №. 3. – S. 377-396.
10. Tsai W. T. et al. Service-oriented user interface modeling and composition //e-Business Engineering, 2008. ICEBE'08. IEEE International Conference on. – IEEE, 2008. – S. 21-28.
11. Gamma E. Pattern languages of program design 3. – Addison-Wesley Longman Publishing Co. – 1997. – S. 79–85.
12. Kaminski P., Müller H., Litoiu M. A design for adaptive web service evolution //Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems. – ACM, 2006. – S. 86-92.

DOI: 10.7256/2305-6061.2016.2.18627