Igor ANDRUSHCHAK[1], Andriy SVERSTIUK[2], Nazar MILYAN[3]

Opiekun naukowy: Vasyl MARTSENYUK[4]

ARCHITEKTURA MIKROSERWISOWA WYBRANEGO ROZWIĄZANIA DLA MEDYCYNY RATUNKOWEJ

Streszczenie: Artykuł jest poświącony prezentacji podejścia mikroserwisów w celu opracowania systemu informacyjnego dla medycyny ratunkowej. Realizacja jest oparta na szablony Spring, Spring Boot oraz Spring Cloud.

Słowa kluczowe: medycyna ratunkowa, mikroserwisy, spring

MICROSERVICES ARCHITECTURE OF ONE SOLUTION FOR EMERGENCY MEDICINE

Summary: The work is devoted to presentation of microservices approach for development of information system for emergency medicine. Implementation is based on usage of spring, Spring Boot and Spring Cloud frameworks.

Keywords: emergency medicine, microservices, spring framework

1. Introduction

A variety of approaches and techniques are used when developing information systems for medical practice and research [1-17].
Primarily they are based on so called monolithic architecture. In such case you are developing a server-side enterprise application. It must support a variety of different clients including desktop browsers, mobile browsers and native mobile applications [18].

[1] Narodowy Uniwersytet Techniczny w Łucku, wydział, specjalność: analiza systemowa, email 9000@lntu.edu.ua
[2] Narodowy Uniwersytet Medyczny w Tarnopolu, specjalność: modelowanie matematyczne, email sverstyuk@tdmu.edu.ua
[3] Narodowy Uniwersytet Techniczny w Tarnopolu, specjalność: technologie informacyjne, email nazar.milyan@gmail.com
[4]prof. dr hab., Akademia Techniczno-Humanistyczna w Bielsku-Białej, wydział budowy maszyn i informatyky, email vmartsenyuk@ath.bielsko.pl

In turn Microservice is a paradigm that serves for organization and usage of distributed services that can have different proprietors. The basic ideas of this architectural approach was stated by Martin Fowler in 2014 in [19].
Microservices allow large systems to be built up from a number of collaborating components. It does at the process level what a lot of frameworks (e.g., Spring) has always done at the component level: loosely coupled processes instead of loosely coupled components.
Much technologies and the protocols that are created to realization of business processes [20] and their support are included in the content of widely understoodable microservices architechture. Those technologies together create powerfull instruments for
- implementation of business processes,
- opening for access to the client for different type of services through a network,
- seeking out of services (UDDI Universal Description, Discovery of and of Integration);
- uses of services through a client,
- determinations of business processes with help of languages of determination of flow of problems and creations of complex services.
Mentioned above possibilities of microservices technology and in particular arrangement of services in greater processes are as background of application of microservices to the construction of the system of medical information services. However at the market there is a shortage of such solutions now.
Composition of services in microservice architecture results in combining of separate services (WS, Web Service) in a structure named by a process that describes the algorithm of implementation of series of services. In order to do it, possession of the detailed information on motion of process is needed (before it will be determined).
In case of emergency medicine service the detailed determination of tasks is not possible. Reason is unpredictability of motion of incident and also factors that influence on flow of possible treatment scheme:
number and state of health of patients;
variety of services that belong to many health establishments;
dynamics of patient state can change in the process of implementation of treatment;
accessability (primarily distance) to healthcare establishments for patient.
We can divide those factors into two groups:
1) information on state of patient previously known. It should be known from the point of view of healthcare, e.g. patient passport data, health assessment, previous histories of diseases and so on,
2) exceptional/change of certain factors. It can be instant change of well-known or unknown previously factor, e.g.: fracture, blooding, emergency etc.
It all causes that the detailed setting by default, how procedure of emergency medicine should look like, is impossible. Moreover, settings are not possible usually, how must be conducted healthcare in situation, when this situation will arise up already. The reason of that is a typical shortage of sufficient data during undertaking of action, since at the beginning of healthcare as a rule we don't have possibility to deliver accordingly exact data [21].
The objective of this paper is to present the way of application of Microservices architecture for the problem of development of information system for emergency medicine.

Potential problems that touch an emergency medicine and idea of construction of the system that basing on the paradigm of Microservice forms separate healthcare services into one coordinated healthcare process, will be presented in the paper. The offered solution means to determine dynamically process without detailed definition from point of view of the whole action. When basing the behaviors on typical charts, the system executes certain introductory steps. During their execution, additional data on the basis of the system for determination of further actions are gotten. As far as progress of emergency medicine action gets each time more detailed data dealing with certain case and necessary actions, a process is being specified in the process of implementation.

2. Materials and Methods

Conception of solution of problem of emergency medicine. Consider that on a certain street in city accident has happend. In accident a dozen of car passengers damaged. Many people feel damages and need emergency medical assistance.

From the point of view of rescue services a problem is difficult for the solution, because information about a case is usually inexact on this stage: it is unknown, how many people are injured and how their damages are serious, how many coaches are needed on the place of accident, how many places in a hospital needed to be prepared to give a help for injured.

We need fast and well-coordinated rescue action. Human factor can lead to errors under such circumstances, which can influence on saving lifes of injured ones. In this situation it is the best to dispose of the certain system able to manage a rescue action, liquidating the errors related to the emotions, by work in stress and necessity of the rapid reacting.

Possibility of support of rescue services is covenient here through the computer system that has an access to all well-known information about a case, that is capable efficiently to plan actions on the basis of all well-known data, is able dynamically to adjust actions to the changable conditions and also to coordinate the actions of elements of different services.

Requirements for information system. In ideal case system that would solve the above-mentioned problems that arise in rescue medical service should be able:

to give independence to the different elements that are his constituents;

to do possible a suggestion of its own services in the system;

to do possible composition of component services into more complicated one such as a complex service for injured person.

Main principle is opening for access of actions of medical services as network services of web-service at application of paradigm of microservices.

The system implements its task through a construction of a skeleton of process on the initial stage. Then as far as the flow of additional data will grow gradually, it allows corresponding adaptation of actions to the queries for concrete case.

Basic services of emergency medicine. For the purpose of development of the system we need to determine the basic complete set of services that will be able to serve to creation of arbitrary rescue action. It seems that determining a complete set of services in emergency medicine is impossible, taking into account a fact that together with development of medicine there will appear new services that should be taken into

account during composition. The reasoning requires to determine certain standard of description of services, that enables development of new ones in future instead of attempts of determination of complete base that is necessary for system running. Examples of services that will be token into account when developing rescue action are:

Arrival of ambulance to the place of incident (ArrivalToPlaceOfIncident) is simple service that means arrival of ambulances to the given place of incident. It includes arrival to the place of case; implementation of assessment of injured person; delivering primary care; preparing report including data of injured persons in details.

Hospitalization of injured persons (HospitalizationOfPatient) is service that consists of the acceptance of patient to the hospital and the delivering corresponding help to him.

Transportation of patient (TransportationOfPatient) is service implying the transportation of injured patient to the hospital.

3. Results

In this part of the article there will be presented an example of problem solution of emergency medicine action using basic set of services mentioned above (Fig.1).

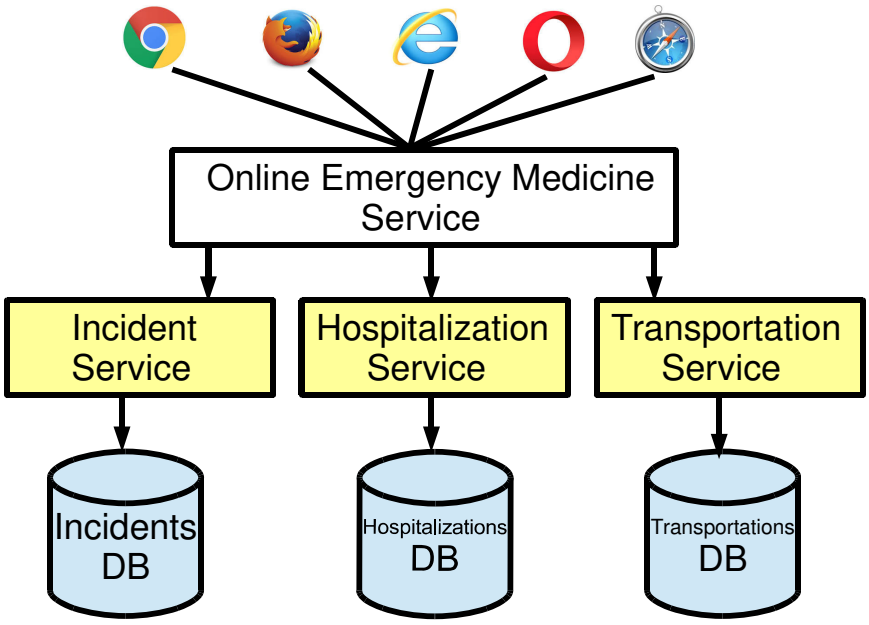

Figure 1. Problem solution of emergency medicine action using basic set of microservices

Let's consider that a travelling accident happened and casual witness is calling on a line. Operator of line inputs the system data concerning the case, after that the system begins processing the case immediately.

We imagine an online emergency medicine service with separate microservices for arrival-to-place-of-case, hospitalization-of-patient and transportation-of-patient:
Inevitably there are a number of moving parts that we have to setup and configure to build such a system. How to get them working together is not obvious - we need to have good familiarity with Spring Boot since Spring Cloud leverages it heavily, several Netflix or other OSS projects are required and, of course, there is some Spring configuration "magic" [23].
Service Registration. When you have multiple processes working together they need to find each other. If you have ever used Java's RMI mechanism you may recall that it relied on a central registry so that RMI processes could find each other. Microservices has the same requirement.
The developers at Netflix had this problem when building their systems and created a registration server called Eureka ("I have found it" in Greek). Fortunately for us, they made their discovery server open-source and Spring has incorporated into Spring Cloud, making it even easier to run up a Eureka server. Here is the complete discovery-server application:

```
@SpringBootApplication
@EnableEurekaServer
public class ServiceRegistrationServer {

  public static void main(String[] args) {
    // Tell Boot to look for registration-server.yml
    System.setProperty("spring.config.name",  "registration-
server");
    SpringApplication.run(ServiceRegistrationServer.class,
args);
  }
}
```

Spring Cloud is built on Spring Boot and utilizes parent and starter POMs. The important parts of the POM are:

```
    <parent>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-parent</artifactId>
        <version>_Brixton_.RELEASE</version>   <!-- Name of
release train -->
    </parent>
    <dependencies>
        <dependency>
            <!-- Setup Spring Boot -->
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <!-- Setup Spring MVC & REST, use Embedded Tomcat
-->
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
```

```
            <!-- Spring Cloud starter -->
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter</artifactId>
        </dependency>

        <dependency>
            <!-- Eureka for service registration -->
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka-
server</artifactId>
        </dependency>
    </dependencies>
```

By default Spring Boot applications look for an application.properties or application.yml file for configuration. By setting the spring.config.name property we can tell Spring Boot to look for a different file - useful if you have multiple Spring Boot applications in the same project.
This application looks for registration-server.properties or registration-server.yml. Here is the relevant configuration from registration-server.yml:

```
# Configure this Discovery Server
eureka:
  instance:
    hostname: localhost
  client:  # Not a client, don't register with yourself
    registerWithEureka: false
    fetchRegistry: false

server:
  port: 1111   # HTTP (Tomcat) port
```

By default Eureka runs on port 8761, but here we will use port 1111 instead. Also by including the registration code in my process I might be a server or a client. The configuration specifies that I am not a client and stops the server process trying to register with itself.
Try running the RegistrationServer now. You can open the Eureka dashboard here: http://localhost:1111 and the section showing Applications will be empty.
From now on we will refer to the discovery-server since it could be Eureka or Consul.
Creating a Microservice: Arrival-to-Place-of-Incident-Service. A microservice is a stand-alone process that handles a well-defined requirement.
When configuring applications with Spring we emphasize Loose Coupling and Tight Cohesion, These are not new concepts (Larry Constantine is credited with first defining these in the late 1960s [24]) but now we are applying them, not to interacting components (Spring Beans), but to interacting processes.
In this example, we have a simple Arrival-to-Place-of-Incident (Incident) microservice that uses Spring Data to implement a JPA IncidentRepository and Spring REST to provide a RESTful interface to incident information (Fig. 2). In most respects this is a straightforward Spring Boot application.

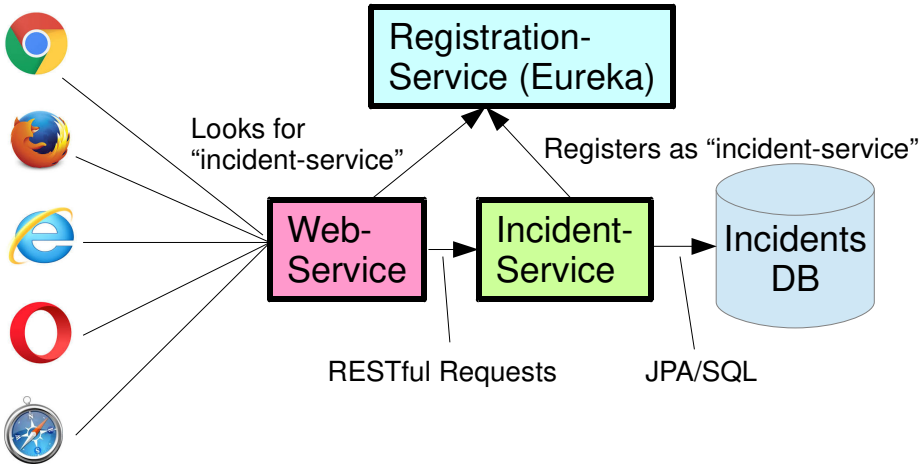

Figure 2. Incident microservice that uses Spring Data to implement a JPA IncidentRepository and Spring REST to provide a RESTful interface to incident information

What makes it special is that it registers itself with the discovery-server at start-up. Here is the Spring Boot startup class:

```
@EnableAutoConfiguration
@EnableDiscoveryClient
@Import(IncidentsWebApplication.class)
public class IncidentsServer {

    @Autowired
    IncidentRepository incidentRepository;

    public static void main(String[] args) {
        // Will configure using incidents-server.yml
        System.setProperty("spring.config.name", "incidents-
server");

        SpringApplication.run(IncidentsServer.class, args);
    }
}
```

The annotations do the work:
@EnableAutoConfiguration - defines this as a Spring Boot application.
@EnableDiscoveryClient - this enables service registration and discovery. In this case, this process registers itself with the discovery-server service using its application name.
@Import(AccountsWebApplication.class) - this Java Configuration class sets up everything else.
What makes this a microservice is the registration with the discovery-server via @EnableDiscoveryClient and its YML configuration completes the setup:

```
# Spring properties
spring:
  application:
     name: accounts-service

# Discovery Server Access
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:1111/eureka/

# HTTP Server
server:
  port: 2222   # HTTP (Tomcat) port
```

Note that this file
Sets the application name as incidents-service. This service registers under this name and can also be accessed by this name.
Specifies a custom port to listen on (2222). All our processes are using Tomcat, they can't all listen on port 8080.
The URL of the Eureka Service process

Run the IncidentsService application now and let it finish initializing. Refresh the dashboard http://localhost:1111 and you should see the INCIDENTS-SERVICE listed under Applications. Registration takes up to 30 seconds (by default) so be patient - check the log output from RegistrationService
For more detail, go here: http://localhost:1111/eureka/apps/ and you should see something like this:

```
<applications>
    <versions__delta>1</versions__delta>
    <apps__hashcode>UP_1_</apps__hashcode>
    <application>
        <name>INCIDENTS-SERVICE</name>
        <instance>
            <hostName>autgchapmp1m1.corp.emc.com</hostName>
            <app>INCIDENTS-SERVICE</app>
            <ipAddr>172.16.84.1</ipAddr><status>UP</status>
            <overriddenstatus>UNKNOWN</overriddenstatus>
            <port enabled="true">3344</port>
            <securePort enabled="false">443</securePort>
            ...
        </instance>
    </application>
</applications>
```

Alternatively go to *http://localhost:1111/eureka/apps/INCIDENTS-SERVICE* and see just the details for IncidentsService - if it's not registered you will get a 404.

4. Conclusions

In the work an innovative approach to construct business process is presented. In opposite to typical application where the process is completely determined before its starting in this case process is not completely determined at the moment when its

running is started. It is implemented using service that can be presented as handler for another process. That service composes process based on data obtained and runs it.
There is some lack of solutions for the problem presented. Although Microservice approach offers tools leading to development of system supporting emergency medicine. An advantage of the system offered is its usage based on mechanism of market. All services are searched through Internet. Moreover any institution can add its own service and in turn to join to the system.
It is not entirely known mechanism of integration of processes in emergency medicine. One of the most promising possibilities is application of Micriservices. Methods of search of appropriate services will be object of future research. In such case an application of ontological descriptions can be solution of the problem. Also the future investigations should be dealing with document formats for exchanging by the system elements, identification of basic services and taking into account economic factors when selecting services for patient.

REFERENCES

1. LYAPANDRA A. S., MARTSENYUK V. P., IGVOZDETSKA. S., SZKLARCZYK R.: Qualitative analysis of compartmental dynamic system using decision-tree induction, in Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) 2015 IEEE 8th International Conference on (Volume 2), 2015, pp. 688–692.
2. MARTSENYUK V. P., ANDRUSHCHAK I. Y., KUCHVARA O. M.: UML-modeling of Decision Support System for Medical Research, Med. Informatics Eng., no. 2, pp. 27–34, 2015.
3. MARTSENYUK V. P., КРАВЕЦЬ Н. О., SEMENETS A. V., ВАКУЛЕНКО Д.В., СВЕРСТЮК А. С., КЛИМУК Н. Я., САРАБУН Р. О., КУЧВАРА О. М.: Про підходи до впровадження emr-систем в галузі охорони здоров'я україни, in "Здобутки клінічної та експериментальної медицини" : матеріали підсумкової науко- во-практичної конференції, присвяченої пам'яті ректора чл.-кор. НАМН України, проф. Л.Я. Ковальчука (Тернопіль, 17 червня 2015р., 2015, pp. 259–260.
4. NAKONECHNY O. H., MARTSENYUK V. P., ANDRUSHCHAK I. Y.: Методи прийняття рішень, оптимізації та керування в системі підтримки медичних досліджень, in XXV International Conference "Problems of Decision Making under Uncertainties (PDMU-2015)" Abstracts. May 11-15, 2014, Skhidnytsya, Ukraine, 2015, pp. 12–13.
5. SEMENETS A. V., MARTSENYUK V. P.: On the CDSS platform development for the open-source MIS OpenEMR, Med. Informatics Eng., no. 3, pp. 22–40, Oct. 2015.
6. MARTSENYUK V. P., ANDRUSHCHAK I. Y.: Розробка клінічної експертної системи, що грунтується на правилах, методом послідовного покриття, Наукові праці. Комп'ютерні технології, vol. 237, no. 225, pp. 5–10, 2014.
7. MARTSENYUK V. P., ANDRUSHCHAK I. Y., STAKHANSKA O. O.: Розробка експертних систем на основі технології Data mining, in Здобутки клінічної та експериментальної медицини. Збірник матеріалів підсумкової науково-практичної конференції. 21 червня 2014 року, 2014, pp. 141–142.
8. MARTSENYUK V. P., MARTSENYUK O. M., ANDRUSHCHAK I. Y.: Mathematical tools for decision support system of medical system research under uncertainties, in XXIV International Conference "Problems of Decision Making under

Uncertainties (PDMU-2014)" Abstracts. September 1-5, 2014, Chesky Rudolets, Chech Republic, 2014, pp. 11–13.
9. NAKONECHNY O. H., MARTSENYUK V. P., ANDRUSHCHAK I. Y.: Інформаційні технології прийняття рішень, оптимізації та керування в системних медичних дослідженнях. Lutsk: ЛНТУ, 2014.
10. MARTSENYUK V. P., SELSKYY P. R.: Ефективність використання інформаційних та телемедичних технологій на первинному рівні надання медичної допомоги, in Матеріали науково-практичної конференції з міжнародною участю "Інформатизація реабілітаційного процесу," 2013, pp. 66–67.
11. MARTSENYUK V. P., SELSKYY P. R.: Ефективність використання телемедичних технологій для покращення якості лікувально-діагностичної роботи на первинному рівні, Актуальні питання фармацевтичної і медичної науки та практики, vol. 12, no. 3, pp. 53–54, 2013.
12. MARTSENYUK V. P., SELSKYY P. R., SEMENETS A. V.: Розробка і впровадження інформаційної системи запису (самозапису) пацієнтів на консультацію до фахівців університетської лікарні, Український журнал телемедицини та медичної телематики, vol. 11, no. 2, pp. 173–178, 2013.
13. MARTSENYUK V. P., ANDRUSHCHAK I. Y.: Про концептуальну модель системи підтримки рішень в системних медичних дослідженнях, in XIX International Conference "Problems of Decision Making under Uncertainties (PDMU-2012)" Abstracts. April 23-27, 2011, Mukachevo, Ukraine, 2012, pp. 162–163.
14. MARTSENYUK V. P., ANDRUSHCHAK I. Y., GANDZYUK N. M., KLYMUK N. Y., KUCHVARA O. M., MAYHRUK Z. V.: Decision Support System for Medical System Research," in XX International Conference PDMU-2012 Problems of Decision Making under Uncertainties Proceedings - Applied Papers, University., E. Hajkova, J. Michalek, O. G. Nakonechny, and J. Neubauer, Eds. Brno: Publishing office of the University of Defence, 2012, pp. 123–128.
15. MARTSENYUK V. P., ANDRUSHCHAK I. Y.: Програмне середовище підтримки системних фармакокінетичних досліджень: підхід на основі Web-технологій, Штучний інтелект, no. 3, 2009.
16. NAKONECHNY O. H., MARTSENYUK V. P., BARANYUK I. O., SVERSTYUK A. S.: Про програмно-технічний комплекс підтримки наукових медичних досліджень, in Медичні технології і вища освіта: Матеріали I Всеукраїнської науково-практичної конференції. Луцьк, 28 травня 2004 р., 2004, pp. 92–97.
17. MARTSENYUK V. P., SEMENETS A. V., SVERSTYUK A. S., KOVALCHUK O. Y., KRAVETS N. O.: Про інформаційну модель інтелектуальної медичної бази даних, in Збірка тез доповідей учасників Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених "Комп'ютери. Програми. Інтернет. 2003" (21-23 квітня 2003 р., м. Київ), 2003, p. 46.
18. Webpage: *http://microservices.io/patterns/monolithic.html*
19. Webpage: *https://martinfowler.com/articles/microservices.html*
20. Webpage: *https://www.infoq.com/articles/soa-healthcare*
21. Webpage: *https://www.pcpcc.org/initiative/primary-care-information-project-pcip*
22. WĄCHOCKI G.: Zastosowanie SOA do celów konstrukcji systemu wspomagającego ratownictwo medyczne, Automatyka 13/2 (2009), 653-661
23. Webpage: *https://spring.io/blog/2015/07/14/microservices-with-spring*
24. Webpage: *https://en.wikipedia.org/wiki/Cohesion_%28computer_science%29*