International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 8 (2017), pp. 2161-2174

http://www.ripublication.com

An Innovative Approach for Cross Functionalities Information Services – A connecting technology

K L Jawaria and Sourabh Taletia
STD & HoD(TPS & SSS)
4th Floor, NIC Hq , New Delhi-110 003

Abstract

In traditional system single application at server side, any change in application will have to discontinue for some time and then load as a fresh. So whenever, there is change in any web application, we have to re-load the new application and system has to keep shutdown for some time. Web Service is a connection technology, a way to connect services together into a Service Oriented Architecture (SOA). As new agile-development practices such as release automation and continuous integration emerged, and new container-based application deployment technology approaches that fully exploit this paradigm shift—commonly called DevOps. The most prominent example of such approach is microservices.

Index Term: API, MSA, WS, SOA, REST, PHP, JSON, SOAP

I. INTRODUCTION

An API stands for Application Programming Interface and does not require an application to be online. A piece of code which does a specific task and shows certain properties which we call the state of the piece of code. This kind of piece of code when put together in one single file is called an Application Programming Interface.

Microservices is a new software architecture. It is based on web services. But it can be any service implemented as an independent feature that has its own database and

can be deployed independently. Microservices and Web Services are two different aspects of Application Development interface, which can be differentiated from the diagram given below. Although "micro" in Microservices, the basic concept is that each service performs a single function.

Many Internet-scale services, including content delivery applications, e-commerce websites, and even governmental digital services, have evolved from monolithic applications to a micro service-based architecture.

II. CHALLENGES IN WEB SERVICES

While using web services for desired information, the requestor and publisher should authenticate and authorize their callers. Unauthorized Access is when a person who does not have permission to connect to or use a system gains entry in a manner unintended by the system owner. The popular term for this is ¯**Hacking**‖.

A. SOA is an architectural pattern in which *application components* provide services to other components. However, in SOA these components *can belong to the same application*. On the other hand, in microservices architecture each component *works as an independent service.*

For us service orientation means encapsulating the data with the business logic that operates on the data, with the only access through a published service interface. The APIs access database with authentication and allowed from outside the service, and there's no data sharing among the services."

B. What Is a Service?

A system supplying a public need.‖ Software services should serve the needs of one or more internal or external client applications. An interface of such desired services should be designed from a client perspective, optimizing for maximum usefulness for its API consumer(s). While ideally every service is different as it optimizes for its clients' use cases, it can generally be said that a service consists of a piece of functionality and its associated set of data. When, a service is an authoritative and definitive source for the current state of the data it holds and vends, then we say a service a system of record.

An example of services where this system-of-record test does not pass could be a service that aggregates data and functionality from other services, which in turn could themselves be the systems of record for this data i.e. an attempt to improve the usability or performance of the entire system.

C. What is Web Service?

Web Service means Connection Techncology : Multiple Unique Services are communciating together. Heterogeneous Database can interact each other with the help of MSA specialized Service. It is API based architecture.

In a Web service we just create a service for an application that can be accessed by web protocols. It is a way to expose the functionality of an application to other application, without a user interface. It is a service which exposes an API over HTTP.

Web Services facilitates to provide the information on demand, however, applications developed in different technologies or platform to communicate with each other through a common format like XML, Jason, etc. Web services are not tied to any one operating system or programming language. For example, an application developed in C# or Java can communicate vice versa.

Working: A web service is a collection of APIs working together to perform a particular task. Web service can be accessed using a transport protocol. HTTP is a far more popular transport protocol to send a request and get a response to and forth from a web service. Using a web service does require us to be online in the first place.

1. All Web services are APIs but all APIs are not Web services.
2. Web services might not perform all the operations that an API would perform.
3. A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication whereas API may use any style for communication.
4. A Web service essentially requires a network for its operation whereas an API does not require a network for its operation.

Hence, Microservices and Web Services are two different aspects of Application Development Architecture, which can be differentiated from its layered architecture and development style as given below.

D. Microservices

Microservices are the architectural evolution of SOA, driven by DevOps practices. The individually deployable services made it easier to apply Continuous Integration / Continuous Deployment.

Monolitic Architecture -> puts all of its functionality into a single process , which is then replicates acrosss multiple servers. It is built as a single unit.The server side application is a monolith - a single logical executable Any changes to the system

involve building and deploying a new version of the server side application. scales by replicating the monolith on multiple servers

Microservices Architecture (MSA): Puts is functionality into different process, which is distributed across many servers as the system requries.Each element of functionlity into a seperate service. Scales by distributing these services across servers replicating as needed. In MSA different teams have a specific focus on, say, UIs, databases, technology layers, or server-side logic, Microservices architecture utilizes cross-functional teams. The responsibilities of each team are to make specific products based on one or more individual services communicating via message bus. Micro Service Architecture (MSA) can be seen as a light weight SOA.

The advantage of MSA is that the services can be scaled independently of other services depending on the load. Also when a service goes down, that does not bring down the entire system. It is a method of breaking large software applications into loosely coupled modules, in which each service runs a unique process and communicates through APIs.

Microservices are designed to cope with failure and breakdowns of large applications. In this MSA multiple unique services are communicating together, it may happen that a particular service fails, but the overall larger applications remain unaffected by the failure of a single module.

Micro Service Architecture (MSA) can be seen as a light weight SOA. Microservices are generally implemented in REST over HTTP protocol. Web services can be implemented in REST, SOAP etc.

III. CROSS FUNCTIONALITIES APPROACH IN INFORMATION SERVICES A. CONCEPTUALIZATION

Let us understand the concept of working with the help of an example of online address proof from Aadhar database in figure-1:

The address verification Web Application is developed in Monolithic Architecture. In Monolithic Architecture application, there is one web service at a time which communicates with web application and database. Hence, web service performs many functional tasks related to database operations.

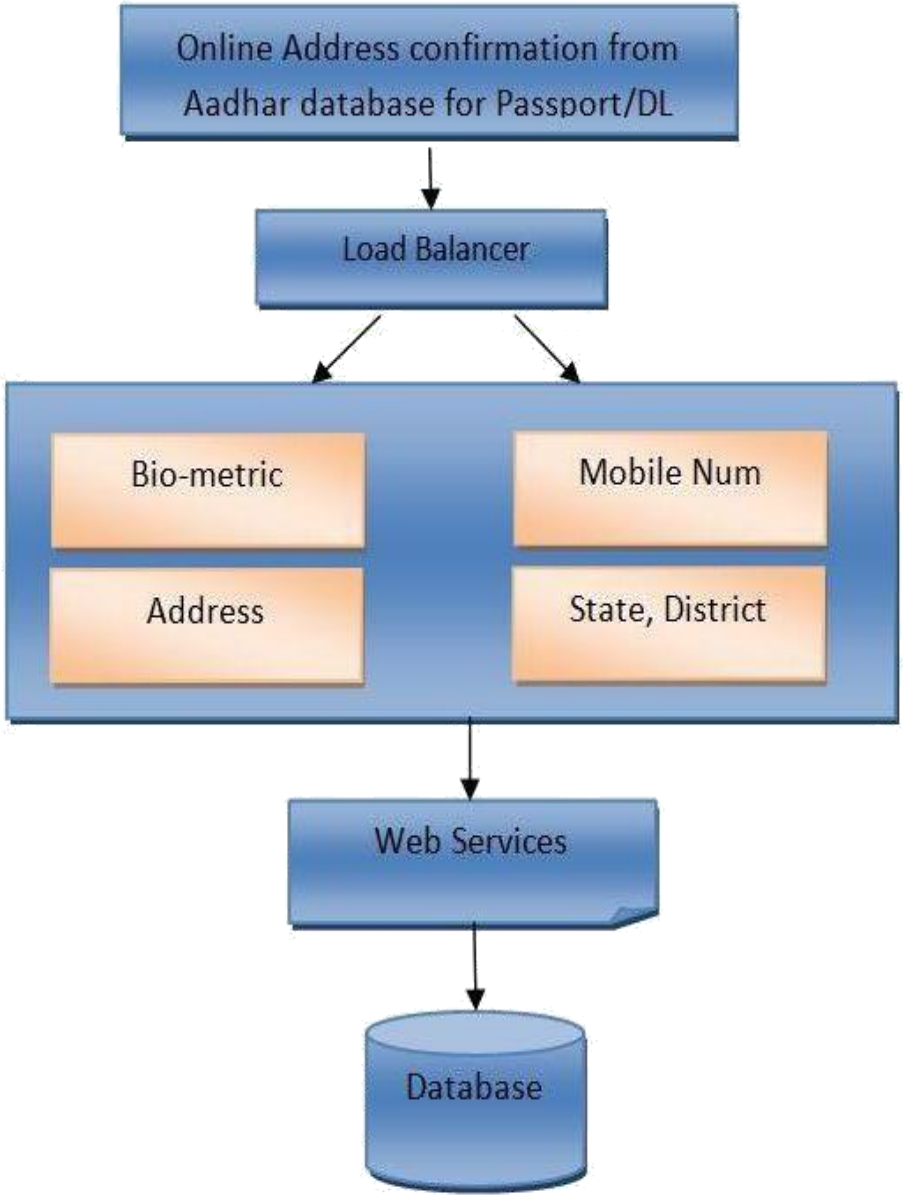


Figure 1: Monolithic Architecture

The address proof Web Application is developed in Microservices Architecture. All the components of the web application are designed and developed in single function and responsible for providing the information exchange services.

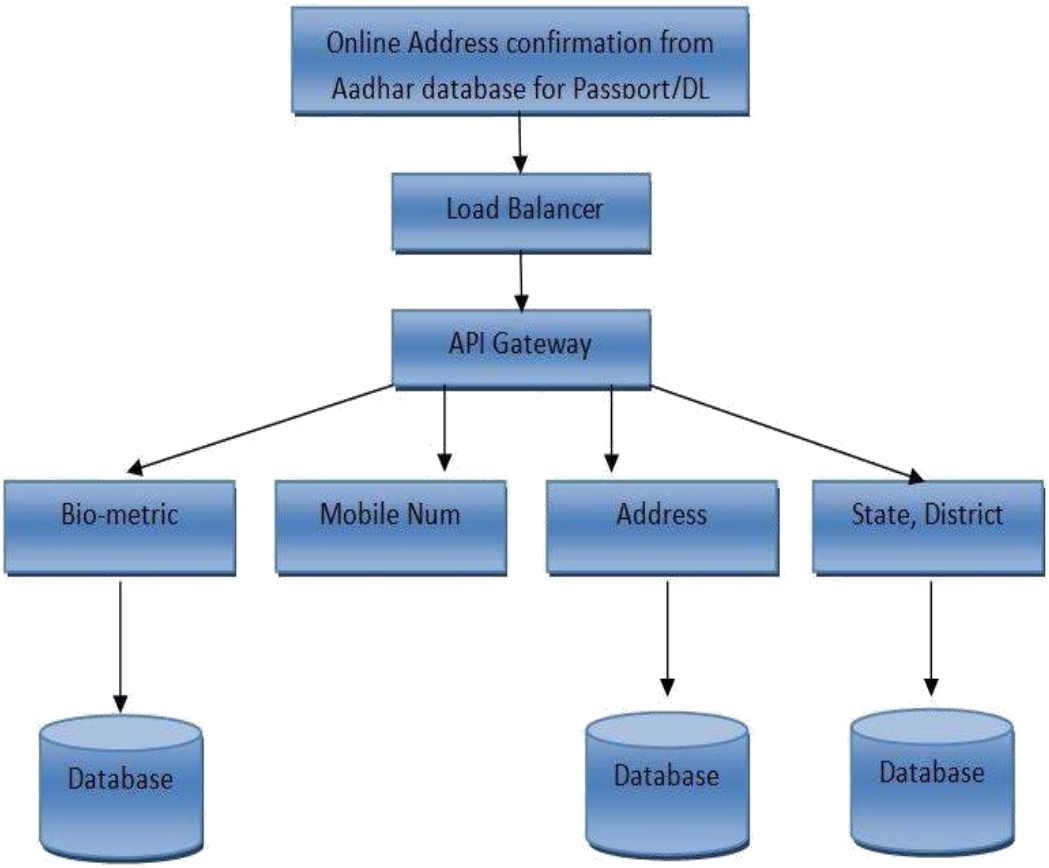
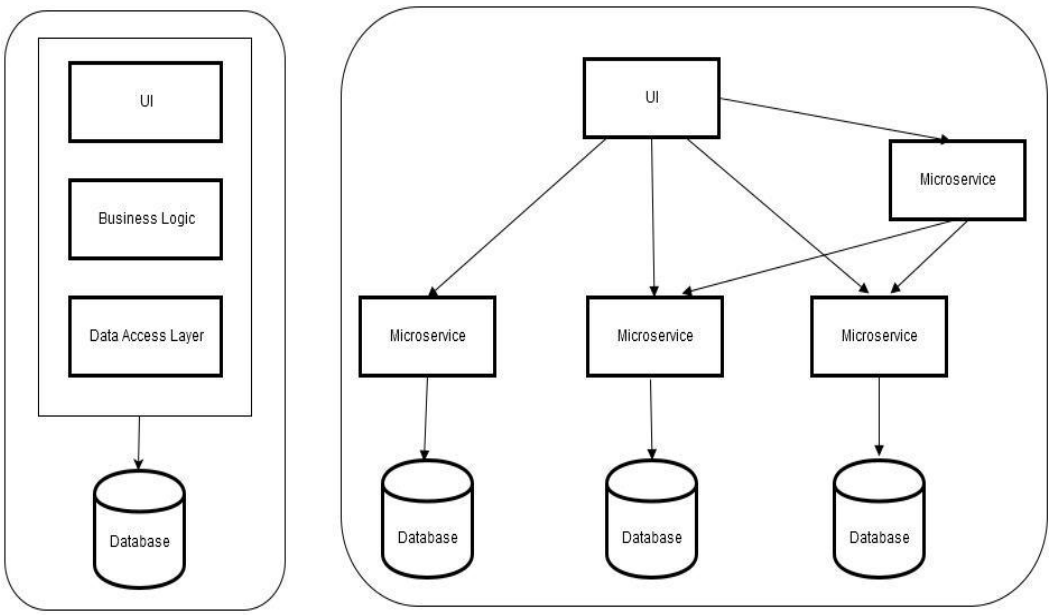


Figure-2: Microservices Architecture

As such there is no restriction in size of web service size. It includes large enterprise apps retrofitted with APIs that too many other apps depended on. For example, one of the largest ecommerce portals, Amazon, has migrated to Microservices. Due to this new architecture, their development team gets few calls even having large variety of applications, including applications that manage the Web Services API as well as the portal, which would have been simply impossible to handle for their old, two-tiered architecture. Applications built as Microservices can be broken into multiple component services and this service can be a Web Service, which should run unique process and then redeployed independently without compromising the integrity of an application.

B. Comparison: Monolithic v/s MS Architecture



Both architecture are about breaking down the monolith backend for greater scalability and maintainability of high traffic and complex web services. While SOA is more enterprise oriented, MSA is simpler in the sense it is easier to do service integration at the receiving end. The monolith backend is partitioned into various services, sometimes as many as hundreds of services using the single responsibility principle. Each service has a bounded context, having its own data model and its own DB. The services

Microservices style is usually organized around business capabilities and priorities. In traditional development approach, where different teams handle specific focus on desired service such as UIs, databases, technology layers, or server-side logic, whereas Microservices architecture utilizes cross-functional teams. The responsibilities of each team are to make specific products based on one or more individual services communicating via message bus.

The main difference is that micro-services have the common trait of decomposing a large application into a set of small, focused, versioned, and loosely-coupled services, which are very inexpensive to spin up and maintain.

Microservices don't necessarily have to be web services, they can serve over file descriptors (pipes, UNIX sockets, etc), through queues, over the network, messages, even email.

The main difference between SOA and microservices lies in the size and scope. SOA

relied on ESB. As the word "micro" suggests, it has to be significantly smaller than what SOA tends to be. Micro service is a smaller independently deployable unit. Beware of very small micro service anti pattern – nano service. A SOA can be either a monolith or it can be comprised of multiple microservices.

Runtime autonomy: in the monolith the overall business logic is collocated. With microservices the logic is spread across microservices. So, all else being equal, it's more likely that a micro service will interact with other microservices over the network--that interaction decreases autonomy. The good news is that to avoid runtime autonomy issues, we can employ techniques such as eventual consistency, event-driven architecture, CQRS, cache (data replication), and aligning microservices with DDD bounded contexts. These techniques are not inherent to microservices, but have been suggested by virtually every author I've read.

D. Benefits:

Whenever, changes are required in application, there won't necessarily be any reason for the project, as a whole, to take more time or for developers to have to wait for budgetary approval before individual services can be improved. Mostly development process focus on projects: a piece of code that has to offer some predefined business value must be handed over to the client, and is then periodically maintained by a team. But in Microservices architecture, a team owns the product for its lifetime.

In a monolithic architecture deployment, a small change meant that the entire monolith needed to be rebuilt and this, in turn, meant that re-builds weren't happening as rapidly as they should.

A Web Service is a service offered by an application to another application, communicating with each other via the Internet.

Service typically provides an object-oriented web-based interface to a database server, utilized by another web server, or by a mobile application, that provides a user interface to the end user.

E. MSA Features:

- It has focused scope that concentrates on a very few functionalities (at times a single one), expressed by well-defined API.
- MSA is modular approach, independently deployable & essentially elevating the rules of software encapsulation and modularity from the program to the deployment unit level.

IV. USE-CASE REPRESENTATION AND SAMPLE CODE

In micro service world a large gigantic software application is break down into individual self-deployable services. Let's discuss about our use case, How Aadhar platform can be leveraged to its full potential using microservices and what would be its benefits.

Aadhar number is an unique id allotted to each person living in India and government is pushing this unique id to be attached to every account. With taking account of India's population and the traffic Aadhar platform would receive, the resource allocation and.

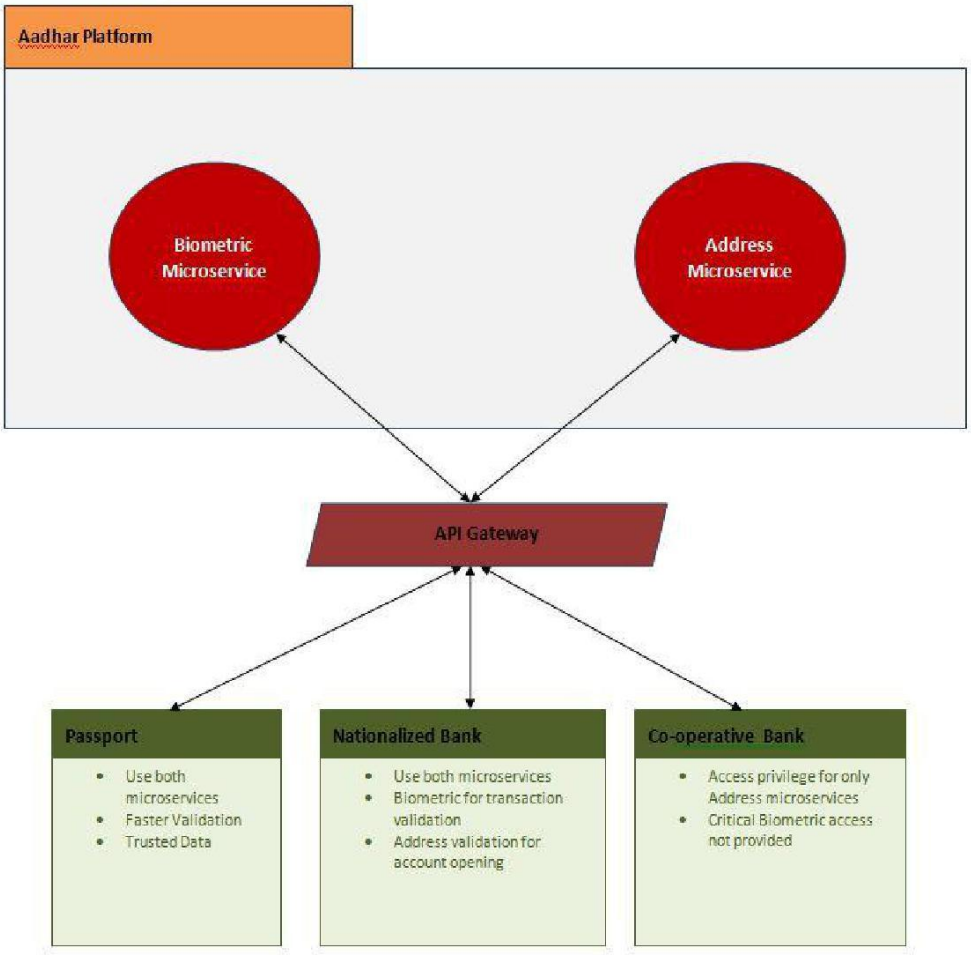
Biometric and Address services can be independent self-deployable services in Aadhar platform.

Each individual service has set of specific domain knowledge people which leverage the quality of service. Biometric service will have resources that better understand the how biometric works and Address services resources better know about the best algorithm and data structures to model & serve address.

Granting access privileges becomes much easier with this approach. Biometric service resources better understand their client and according to client they can provide the access level and grant access to some of the service and hide some. For e.g. The Nationalized Bank would need Biometric for validating transactions and Passport office can use Biometric to validate the person. On the other hand, Co-operative bank are not that much trustworthy to share biometric information hence they can only be provided with Address Service.

Address Service can provide the details of current and past addresses of a person. This could be helpful to lot of verticals. The service team when works individually on one goal it could deliver more features out which would help eliminating lot of fraud.

Address Service is more common used services hence it would need more servers to run. In Monolithic architecture whole Aadhar software has to be installed on the servers which would unnecessary waste the resource and money. In micro service self-deployable container based service can be anytime deployed and brought down, hence we only need to assign more servers to Address service, Biometric service is niche and only used by certain customers hence will need less computes. This design saves us lot of resources and money.



Biometric Microservices

GET : /biometric/person/{aadhar_number} /fingerprint : returns finger print hash
GET : /biometric/person/{aadhar_number} /retina : returns retina hash
GET : /biometric/person/{aadhar_number} : returns list of fingerprint hash and retina hash

POST : /biometric/person/{aadhar_number}/fingerprint/add : returns success/error
POST : /biometric/person/{aadhar_number}/retina/add : returns success/error

PUT : /biometric/person/{aadhar_number}/fingerprint/update : returns success/error
PUT : /biometric/person/{aadhar_number}/retina/update : returns success/error

DELETE : /biometric/person/{aadhar_number}/fingerprint/delete : returns success/error
DELETE : /biometric/person/{aadhar_number}/retina/delete : returns success/error
DELETE : /biometric/person/{aadhar_number}/delete : returns success/error

Address Microservices

POST : /address/person/{aadhar_number}/valid
POST : /address/person/{aadhar_number}/update

GET : /address/person/{aadhar_number} : returns current address
GET: /address/person/{aadhar_number}/city : returns current city
GET: /address/person/{aadhar_number}/state : returns current state
GET: /address/person/{aadhar_number}/history : returns previous address of the person

:
:
:.... Other endpoints

API Gateway

Microservice Name	Host Url	Client Access
Biometric	https://api.aadhar.com/	Passport, Nationalized bank
Address	https://api.aadhar.com/	Passport, Nationalized and Cooperative Bank

Each client who uses microservices needs to get access approval from API Manager/Gateway.

Now let's see for example how does Passport client looks like.

Passport client has 2 classes Address Helper and Biometric Helper.

```
public class AddressHelper {

    public boolean isValidAddress( AadharNumber aadharNumber , Address address) {

        boolean isValid = call POST https://api.aadhar.com/address/person/{aadhar_number}/valid
        if (isValid == false) {
            return Exception("Address does not matches with the person");
        }
        return true;

    }
    public List<Address> getPreviousAddress() {
        // Add previous addresses related to the person
        List<Address> addresses = call GET https://api.aadhar.com/address/person/{aadhar_number}/history
        // Add current address
        addresses.add(call GET https://api.aadhar.com/address/person/{aadhar_number});

        return addresses;
    }
}
```

```
public class BiometricHelper {

    public boolean isBiometricAvailable() {
        boolean isAvailable = Size (call GET https://api.aadhar.com/biometric/person/{aadhar_number}) > 0
        return isAvailable;
    }

    public boolean isValidFingerPrint(AadharNumber aadharNumber, FingerprintData data) {
        Object fingerPrintHash = call GET https://api.aadhar.com/biometric/person/{aadhar_number}/fingerprint
        if (fingerPrintHash == data) return SUCCESS;
        else return ERROR;
    }
    public boolean isValidBiometric();
    public void addFingerPrint(Fingerprint data);
    public void addRetina(Retina data);

}
```

V. CONCLUSION

Microservices are a trendy and highly popular topic in both industry and the scientific community recently, their use has increased dramatically owing to their suitability for building highly available, fault-tolerant, and scalable software by using DevOps practices and container based technologies.

In short, microservices are a realization of the service-oriented architecture (SOA) style for developing software comprising small independent services, each running in its own process and communicating through lightweight mechanisms, often using an HTTP resource API. Microservices facilitates to write services in any programming languages and can use any standard database technologies. MSAs are built around separate business capabilities, independently deployable, scalable by fully automated deployment machinery and with a minimum of centralized management.

REFERENCES

[1] Álvarez, P. et al. (2003) Guelfi, N. et al. (Eds.): A Java Coordination Tool for

[2] Web-service Architectures: The Location-Based Service Context, FIDJI, LNCS 2604, Springer-Verlag Berlin Heidelberg, pp.1–14.

[3] Benatallah, B. et al. (2004a) Atluri, V. (Ed.): On Automating Web services Discovery, Received: December 15, 2002/Accepted: September 15, Published online: February 6, Springer-Verlag.

[4] Benatallah, B., Dumas, M., Fauvet, M-C., Rabhi, F.A. and Sheng, Q.Z. (2002) Overview of Some Patterns for Architecting and Managing Composite Web Services, ACM SIGecom Exchanges, August, Vol. 3, No. 3, pp.9–16.

[5] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and

Orchard, D. (2004) Web Services Architecture, W3C Working Group Note 11, February, W3C Technical Reports and Publications, http://www.w3.org/TR/ws-arch/.

[6] Freund, T. and Storey, T. (2002b) Transactions in the World of Web Services, Part 2. An Overview of WS-transaction and WS-coordination, http://www-106.ibm.com/developerworks/ web services/library/ws-wstx2/. A survey on web services composition

[7] The arrowhead approach for SOA application development and documentation Fredrik Blomstedt; Luis Lino Ferreira; Markus Klisics; Christos Chrysoulas; Iker Martinez de Soria; Brice Morin; Anatolijs Zabasta; Jens Eliasson; Mats Johansson; Pal Varga

[8] Neuman, S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media (2015)

[9] Fairbanks G, Keeling M (2015) Microservices workshop at SEI SATURN 2015. https://github.com/michaelkeeling/SATURN2015-Microservices-Workshop. Accessed 29 June 2016

[10] Fowler M (2014) Microservices prerequisites. Accessed 29 June 2016- http://martinfowler.com/bliki/MicroservicePrerequisites.html.

[11] Giamas A (2016) From monolith to microservices, Zalando's Journey. Accessed 29 June 2016- http://www.infoq.com/news/2016/02/Monolith-Microservices-Zalando.

[12] Hüttermann M (2012) DevOps for developers (expert's voice in Web development), 2012th edn. Apress, New York City

[13] Jones S (2014) Microservices is SOA, for those who know what SOA is. http://service-architecture.blogspot.ch/2014/03/microservices-is-soa-for-those-who-know.html. Accessed 29 June 2016

[14] Lewis J, Fowler M (2014) Microservices—a definition of this new architectural term. http://martinfowler.com/articles/microservices.html. Accessed 29 June 2016

[15] Little M (2015) SOA versus microservices? http://www.infoq.com/news/2015/02/special-microservices-mark-litle. Accessed 29 June 2016

[16] Loftis H (2016) Why microservices matter. https://blog.heroku.com/archives/2015/1/20/why_microservices_matter. Accessed 29 June 2016

[17] 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA) http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html. Accessed 29 June 2016

[18] Newman S (2015) Building microservices, 1st edn. O'Reilly Media, Sebastopol, CAGoogle Scholar

[19] Richardson C (2014) Microservices: decomposing applications for deploy ability and scalability. https://www.infoq.com/articles/microservices-intro. Accessed 29 June 2016

[20] Rotem-Gal-Oz A (2014) Services, microservices, nanoservices—oh my! http://arnon.me/2014/03/services-microservices-nanoservices/. Accessed 29 June 2016

[21] Strumpflohner J (2015) Notes and thoughts on Martin Fowler's talk about microservices at XConf. http://juristr.com/blog/2015/01/notes-microservices-fowler-xconf/. Accessed 29 June 2016

[22] Virdell, M. (2003) Business Processes and Workflow in the Web Services World, http://www-106.ibm.com/developerworks/webservices/library/ws-work.html, January.

[23] Wähner K (2015) Do good microservices architectures spell the death of the enterprise service bus? https://www.voxxed.com/blog/2015/01/good-microservices-architectures-death-enterprise-service-bus-part-one/. Accessed 29 June 2016