# Research Methodology

## Experiment

This research analyses three different languages like Java, which is widely used for the development of microservices [(Glen, 2018)](https://www.zotero.org/google-docs/?tyyVht), Jolie a language specially designed for the distributed system to support the service oriented architecture [(Montesi et al., 2014)](https://www.zotero.org/google-docs/?USII7s) and Ballerina, the newly developed languages for microservices [(WSO2 Inc, n.d.)](https://www.zotero.org/google-docs/?QOOttH). In order to answer what programming language is the best it is important to analyse, empirically, the programs written in that language [(Nanz & Furia, 2015)](https://www.zotero.org/google-docs/?9JwkB3). The experiments performed in the controlled fashion help to understand the properties of the programming language and most reliable data is obtained as the outcome of the controlled experiment [(Nanz et al., 2013)](https://www.zotero.org/google-docs/?5WuUln).

### Control Variable

To understand the impact of the integration on these programming languages, my research uses the controlled experiment methodology. The microservices program for all the three languages is developed by me and to ensure the quality of the code, the programs written in Java are as per the tutorial guide given by Oracle. These Java programs serve as the baseline for the programs in other two languages and these programs are replicas of the Java programs in terms of the functionality. Thus, enforcing the high control at the development stage of the experiment. Microservices and the containers are a natural pair that helps the former to achieve higher levels of modularity, code reuse, reproducibility and fine-grained scaling of a distributed system [(Stubbs et al., 2015)](https://www.zotero.org/google-docs/?CaX0mc). Container technology, and Docker in particular, is making a profound impact on microservices. Thus, making Docker the first choice for the research to run the microservices. The control for deployment stage is achieved by running all the services in the common Docker environment that have the same memory allocated for all the services. Finally, the platform that invokes the service will be the Postman client and the shell script that will be common for all the services. Thus, the control for this experiment has been achieved at all the different stages of software development life cycle - development, deployment and execution.

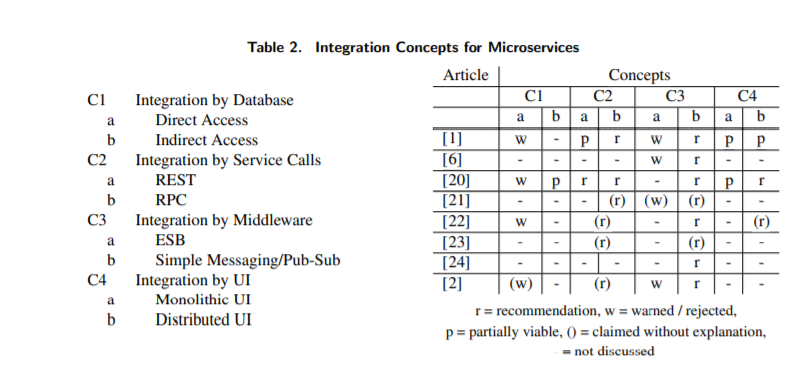
### Independent Variable

Apart from the control, there are two more variables important for an experimental approach [(Jackson, 2014)](https://www.zotero.org/google-docs/?07ch7o). These variables are independent and dependent variables. They help to understand their impact and measure it for each programming language, respectively. In the experimental process, certain factors are selected and deliberately varied in a controlled manner to understand their effects on the response of interest [(Wang & Wan, 2009)](https://www.zotero.org/google-docs/?1Kn8Dm). These are called independent variables. The research focuses on the integration of the microservices, thus the effect of integration can be determined by testing the language for different integration technologies and integration data.

#### Integration Technology

As microservices development approach follows smart endpoint and dumb pipeline philosophy, the best strategy for building the microservices from the integration standpoint is to use the technology that is language agnostic and simple to use [(Fowler, 2010; Newman, 2015)](https://www.zotero.org/google-docs/?73ofMZ). According to [(Fowler, 2010)](https://www.zotero.org/google-docs/?XWasZy) request-response or messaging enables to achieve the integration in the simple possible way by making use of the principle like ReST or queues for communication.

The research considers the internal integration of the microservices, that is the communication between the microservices. [Schwarz and Riehle (2020)](https://www.zotero.org/google-docs/?itDyVd) generated a comparative model for different integration designs and this model suggested the technology that is best fit for these kinds of integration. ReST, RPC, and Messaging are the three designs that are highly recommended in these models and my experiment uses the same integration concept to identify the effect of them on the languages selected for comparison.



Two simple services are built and tested for different integration technology. These services are simple services namely hello service and the world service and they communicate with each other using three different integration technologies send across messages in different data formats.

##### Representational State Transfer (ReST)

ReST is the set of principles for the request response based service development and is very popular in the web service development community [(Li & Chou, 2011)](https://www.zotero.org/google-docs/?04tcl8). According to [Richards (2015)](https://www.zotero.org/google-docs/?NzVAtW), microservices architectures rely on REST as their primary remote-access protocol. Others like [Newman (2015)](https://www.zotero.org/google-docs/?qcI9lO) and [Fowler (2010)](https://www.zotero.org/google-docs/?TZxbth) also recommend ReST over HTTP for service to service interactions in the microservices architecture. One of the key things for this research is the use of the standard that is loosely coupled with the technology or the programming language, as that also serves as an important factor for choosing any language for development of microservices from the integration viewpoint. ReST is totally language agnostic and thus testing the language for ReST is important.

##### Remote Procedure Call (RPC)

RPC again is the request response protocol where the request is made by the client to execute on the remote system. The client can make this remote call directly as if it was made from the local object. There are different implementations of RPC, but the one that this research uses considers two factors, being technology agnostic and the implementation recommended for microservices architecture, one of them is gRPC [(Gan et al., 2019; *GRPC*, n.d.)](https://www.zotero.org/google-docs/?HWN3DR). gRPC is an open source framework for remote procedure calls developed by Google. gRPC uses protocol buffers as the service definition language and the format to exchange the data.

##### Messaging

Messaging follows publisher subscriber design protocol, where the publisher service sends the message to a queue or a topic and the subscriber service consumes when the event occurs. Like RPC, there are different implementations and standards for efficient messaging. However, the research will make use of the latest protocol available for messaging called Advanced Messaging Queue Protocol (AMQP) and the implementation used for AMQP is RabbitMQ as it is one of the open source framework available and is also suggested in the official website of AMQP [(*Products and Success Stories | AMQP*, n.d.)](https://www.zotero.org/google-docs/?oL9iid)

#### Integration Data

##### JSON

##### XML

##### Protocol buffers

### Dependent Variable

### Source Code

### Deployables

### Execution

### Output

# References

[Fowler, M. (2010). *Richardson Maturity Model*. Martinfowler.Com. https://martinfowler.com/articles/richardsonMaturityModel.html](https://www.zotero.org/google-docs/?drUn4q)

[Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., Bruno, A., Hu, J., Ritchken, B., & Jackson, B. (2019). An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 3–18.](https://www.zotero.org/google-docs/?drUn4q)

[Glen, A. (2018, July 26). *[DZone Research] Microservices Priorities and Trends*. Dzone.Com. https://dzone.com/articles/dzone-research-microservices-priorities-and-trends](https://www.zotero.org/google-docs/?drUn4q)

[*GRPC*. (n.d.). GRPC. Retrieved September 22, 2020, from https://grpc.io/](https://www.zotero.org/google-docs/?drUn4q)

[Jackson, S. L. (2014). *Research methods: A modular approach*. Cengage Learning.](https://www.zotero.org/google-docs/?drUn4q)

[Li, L., & Chou, W. (2011). Design and Describe REST API without Violating REST: A Petri Net Based Approach. *2011 IEEE International Conference on Web Services*, 508–515. https://doi.org/10.1109/ICWS.2011.54](https://www.zotero.org/google-docs/?drUn4q)

[Montesi, F., Guidi, C., & Zavattaro, G. (2014). Service-oriented programming with Jolie. In *Web Services Foundations* (pp. 81–107). Springer.](https://www.zotero.org/google-docs/?drUn4q)

[Nanz, S., & Furia, C. A. (2015). A comparative study of programming languages in rosetta code. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, *1*, 778–788.](https://www.zotero.org/google-docs/?drUn4q)

[Nanz, S., West, S., & Silveira, K. S. D. (2013). *K.: Benchmarking usability and performance of multicore languages*.](https://www.zotero.org/google-docs/?drUn4q)

[Newman, S. (2015). *Building microservices*. O’Reilly Media, Inc. https://learning.oreilly.com/library/view/building-microservices/9781491950340/titlepage01.html](https://www.zotero.org/google-docs/?drUn4q)

[*Products and Success Stories | AMQP*. (n.d.). Retrieved September 22, 2020, from https://www.amqp.org/product/realworld](https://www.zotero.org/google-docs/?drUn4q)

[Richards, M. (2015). *Microservices vs. Service-oriented architecture*.](https://www.zotero.org/google-docs/?drUn4q)

[Schwarz, G.-D., & Riehle, D. (2020). What Microservices Can Learn From Enterprise Information Integration. *HICSS*, 1–10.](https://www.zotero.org/google-docs/?drUn4q)

[Stubbs, J., Moreira, W., & Dooley, R. (2015). Distributed systems of microservices using docker and serfnode. *2015 7th International Workshop on Science Gateways*, 34–39.](https://www.zotero.org/google-docs/?drUn4q)

[Wang, J., & Wan, W. (2009). Experimental design methods for fermentative hydrogen production: A review. *International Journal of Hydrogen Energy*, *34*, 235–244.](https://www.zotero.org/google-docs/?drUn4q)

[WSO2 Inc. (n.d.). *Ballerina.io*. Retrieved September 6, 2019, from https://ballerina.io/](https://www.zotero.org/google-docs/?drUn4q)