# Methodology

According to Nanz and Furia (2015) a reliable answer to what programming language is the best, can be given only by analyzing empirically with the help of artifacts built in those programming language. In this research, the same approach is used to find out the best programming language from the three chosen languages – Ballerina, Jolie and Java.

To compare the languages, researchers have been using the existing repository like (*Rosetta Code*, n.d.) that has 1059 programs written in 791 languages that is available for the researchers as well as the developers for the reference (Nanz & Furia, 2015) (Nanz et al., 2013). There are other research conducted to compare the programming languages where the researchers have built their own artifacts for comparison based on the benchmark problem statement (Nanz et al., 2013). My research will also use similar experimental methodology with same parameters identified for comparison with additional parameters that focus on the integration aspect of the programming language to answer the research question.

**Finding Parameters -**

The parameters from the linguistic approach for microservices are interfaces or services as they separate the behavior from deployment (Guidi et al., 2017). This can be possible if the programming language treats Interfaces or Services as the first-class entity. Besides Interfaces, Ports and Workflows are also important for microservices. The other factors important for the general purpose programming language is verbosity of the source code, size of the executables and the execution time of the code (Nanz & Furia, 2015).

The parameters identified are measured by running the programs for different integration scenarios.

**Finding Sample Integration Program –**

The research uses the integration scenarios that are identified by the Official website of each programming language provider. For java, Spring community (*Engage with the Spring Community*, n.d.) has listed numerous java sample code for integration that can be found in the GitHub repository maintained by Spring community. This study uses java code as the bases to program in other two languages.

The programs used for comparison in the research will be stored in GitHub and given the public access.

**Comparison of Program –**

After downloading the different programming languages code from the respective repos, these programs are compared at three different stages – source code, binaries and execution. At the source code stage, the program is imported in the Integrated Development Environment (IDE) used for the programming language. Programs at this stage will be compared for verbosity, debug, interfaces/services as the first-class constructs. At the binaries stage, the code will be built to be translated to the binary code that can be run on different operating system. Programs at this stage will be compared for size. At the last stage of execution, the binaries will be executed on different operating system and the parameters measured in this stage are execution time.

To ensure the output derived is due to the programming language alone, the programming for different integration scenario is done in the exact manner as java. Also the other factors like the runtime environment will have the same configuration for all the programming language. Also, the programs will be run in both the Windows and the Linux operating system.

**Parameters**

The research will talk in brief about each parameter that is used for the comparison of the language.

### Verbosity:

Term Verbose means explaining in words more than needed. According to the developer community StackOverFlow, majority of the developers supported that a good code should be easy to comprehend at a glance (Why Is Verbosity Bad for a Programming Language?, n.d.). Also, according to the Gupta (2004), excess verbosity can cause the programmers at the beginner level to miss the conceptual learning in order to get the grasp of the huge code. When a programming language is less verbose it also mean that it is capable to do a task in less number of lines which also saves the development time with respect to writing code (Krishna et al., 2017). Considering all this factors, the line of code for each program will be measured and the languages will be evaluated on the numbers obtained from the experiment.

### Size:

Size of the executable has a great impact on the deployment of the code. Reduced size of executable has several benefits like faster deployment, smaller disk size, small server space, reduced cost of hardware and easy management of code (Cepa, 2005). The factors that influence the size of the code is the use of third-party libraries or the external frameworks to support a development. However, the same can be avoided if the language development environment itself provides those libraries. (How Ballerina  Is Different From Other Programming Languages - DZone Integration, n.d.). In this study the executable built for every language will be evaluated against size of the executables for every integration scenario.

### Execution Time

It is a well-known fact that faster applications are preferred over the slow performing applications. Execution time forms an important parameter to measure the performance of any application. A new technology is always assessed on its performance and execution time is one of the parameter to measure the performance (Pongnumkul et al., 2017). In this research, for every program the execution time will be recorded and compared for all the languages.

### Debug

Debugging is useful in various stages of software development like design where the error made in writing the code is identified, secondly it is also helpful in the later stages of testing. Debugging is also helpful in the error diagnostics in the production environment (Cheng et al., 2017).//TODO: need to identify how to compare on basis of debug.

### Interfaces and Port

Microservices are deployed as the “black boxes” whose implementation details are hidden in order to support modular programming. The details of the functionality of the services is provided by the interfaces. Interfaces describe the set of operations that can be remotely invoked. Thus, interfaces become the first-class citizen of the microservices. Thus, it is important for a microservices languages to provide the set-theoretical operators like union and intersection that can work with the interfaces (Guidi et al., 2017).

Besides Interfaces, ports are important as communication port describes how the services are made available to the network. Each service may be equipped with many ports thus ports should be separate from the implementation of the service. Hence, the language for microservices should provide the capability to separate the port from the implementation of the service (Guidi et al., 2017).

This research will identify what language treats Interfaces and Ports as the first-class citizen. This can be seen from the artifact built in all three languages.

### Graphical View Support

It is a common practise to draw the sequence diagram for the complex integration scenarios (Weerawarana et al., 2018). Also, there are different workflow languages like BPEL being used by the industry experts to simplify the complex integration scenarios (Guidi et al., 2017). Thus, it is helpful for the microservices language to provide the visual support or the workflow development options to support the development of microservices. This research will test the three languages for its capability to support visual designing. This can be seen from the artifact built in all three languages.

**Integration Scenarios**

**Advanced Messaging Queuing Protocol –**

The Advanced Message Queuing Protocol (AMQP) is an open standard for passing business messages between applications or organizations.  It connects systems, feeds business processes with the information they need and reliably transmits onward the instructions that achieve their goals. (<https://www.amqp.org/>). AMQP connects across the organizations, applications on different platform, also the systems do not need to be available simultaneously and can operate over poor network as well.  There are different AMQP technology suppliers, the supplier used in the sample code of java spring-integration is RabbitMQ as it is widely used at different organizations in the integrated systems (<https://www.amqp.org/product/realworld>). The same supplier will be used to test the amqp integration scenario for Ballerina and Jolie.

**Pre-requisites :**

Docker

Running instance of RabbitMQ - <https://www.rabbitmq.com/install.html>

Java 8

Ballerina 1.2.6

**Java -**

The java program for RabbitMQ is available on the official website of RabbitMQ. <https://github.com/rabbitmq/rabbitmq-tutorials/tree/master/java>

The same code is written in Ballerina referring to the sample provided by the official website of Ballerina –

**Java Messaging Service –**

Java - <https://github.com/spring-projects/spring-integration-samples/tree/master/basic/jms>

Ballerina - <https://github.com/ballerina-platform/module-ballerina-java.jms/tree/master/examples>

Jolie – Unavailable

**Rest –**

Java - <https://github.com/spring-projects/spring-integration-samples/tree/master/intermediate/rest-http>

Jolie - Unavailable

Ballerina – Unavailable

**Xml support –**

Java - <https://github.com/spring-projects/spring-integration-samples/tree/master/basic/xml>

XML - <https://ballerina.io/learn/by-example/xml.html>

JSON support –

Java – Unavailable

Ballerina - <https://ballerina.io/learn/by-example/json.html>

Jolie – Unavailable

Database –

Java – <https://github.com/spring-projects/spring-integration-samples/tree/master/basic/jdbc>

Ballerina – <https://ballerina.io/learn/by-example/jdbc-client-crud-operations.html>

Jolie – Unavailable

# References

Cepa, V. (2005). Using Reflection to Reduce the Size of. NET Executables. *J. Object Technol.*, *4*(7), 51–64.

Cheng, C.-T., Lee, Y.-H., Liu, C.-H., & Wu, M.-T. (2017). *Error diagnostic in a production environment*. Google Patents.

*Engage with the Spring Community*. (n.d.). Retrieved July 17, 2020, from https://spring.io/community

Guidi, C., Lanese, I., Mazzara, M., & Montesi, F. (2017). Microservices: A language-based approach. In *Present and Ulterior Software Engineering* (pp. 217–225). Springer.

Gupta, D. (2004). What is a good first programming language? *XRDS: Crossroads, The ACM Magazine for Students*, *10*(4), 7–7. https://doi.org/10.1145/1027313.1027320

*How Ballerina  Is Different From Other Programming Languages—DZone Integration*. (n.d.). Retrieved June 9, 2020, from https://dzone.com/articles/ballerinawhy-it-is-different-from-other-programmin

Inc, W. (n.d.). *Ballerina—Ballerina by Example*. Retrieved July 17, 2020, from https://ballerina.io/learn/by-example/

*Jolie/examples*. (2020). [Jolie]. The Jolie Programming Language. https://github.com/jolie/examples (Original work published 2016)

Krishna, R., Menzies, T., & Layman, L. (2017). Less is more: Minimizing code reorganization using XTREE. *Information and Software Technology*, *88*, 53–66.

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.

Nanz, S., West, S., Da Silveira, K. S., & Meyer, B. (2013). Benchmarking usability and performance of multicore languages. *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 183–192.

Pongnumkul, S., Siripanpornchana, C., & Thajchayapong, S. (2017). Performance analysis of private blockchain platforms in varying workloads. *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 1–6.

*Rosetta Code*. (n.d.). Retrieved July 6, 2020, from http://rosettacode.org/wiki/Rosetta\_Code

*Spring-projects/spring-integration-samples*. (2020). [Java]. Spring. https://github.com/spring-projects/spring-integration-samples (Original work published 2011)

Weerawarana, S., Ekanayake, C., Perera, S., & Leymann, F. (2018). Bringing middleware to everyday programmers with ballerina. *International Conference on Business Process Management*, 12–27.

*Why is verbosity bad for a programming language?* (n.d.). Software Engineering Stack Exchange. Retrieved June 8, 2020, from https://softwareengineering.stackexchange.com/questions/141175/why-is-verbosity-bad-for-a-programming-language