An open source Service Oriented Programming language: Jolie (Java Orchestration Language Interpreter Engine)

Mrs. Kiran M. Shinde
Asst.Professor
Institute of Industrial and Computer Management & Research,
(I.I.C.M.R.), Nigadi, Pune Pin. : 411044 Maharashtra, India
Affiliated to University of Pune, Recognized by AICTE
+9102027657648,
(Mobile) +917721991112
kiranshinde.iicmr@gmail.com
Ms.Prajakta Patil
MCA (pursuing)
Institute of Industrial and Computer Management & Research,
(I.I.C.M.R.), Nigadi, Pune Pin. : 411044 Maharashtra, India
Affiliated to University of Pune, Recognized by AICTE
+917040509082
123prajaktapatil@gmail.com

ABSTRACT:

Service oriented computing is a methodology for developing distributed applications based on services. These Services are the basic software components that use interfaces and include few functions which are called to execute a specific task. The evolving mechanisms are used in service-oriented computing for the composition of services which are used by orchestrators to furnish much complex features where service interactions are there to create higher level business services. Orchestrator are capable of invoking, organizing and coordinating other resources over making use of traditional workflow patterns such as sequencing, parallel composition and choices. WS-BPEL and XLANG are examples of orchestration languages. Micro service is an expanding development method where soft wares are developed by creating independent entities, known as micro services. Micro services are usually used to develop individual applications in order to make it more agile and scalable. Jolie is the first service oriented open source programming language based on the micro services paradigm where the code is embedded in services. From local to remote, one can still switch without modifying the Program or the logic and can distribute a small section of it to execute the same on any machine. This article is aimed at making the awareness about the micro service as an implementation strategy for Service Oriented Architecture and Jolie- a service oriented programming language.

Keywords: SOA, Micro services, Jolie, orchestration, web services

INTRODUCTION

Jolie (Java Orchestration Language Interpreter Engine) is a first programming language open source based on the micro services framework, all components are autonomous services that can be independently distributed and controlled by running parallel processes. Jolie provides an abstract layer; tend to range from TCP/IP sockets to system memory communication among operations that allows utility, to communicate via various media. Each program is a set of services that by sending and receiving messages over a network can connect with other programs. Jolie is recently contributed by a Java language interpreter that can be used on different Operating Systems like Linux, OS X and Windows with formal definitions where the language falls into also which helps to ensure it is mathematically specified for Jolie to execute programs. In experiments, Jolie is used to examine language based methods for creating distributed systems. Under the GNU Library General Public License, the key technology (interpreter and standard library) is released (v2 or later) [2] .Jolie open source project was initiated by Fabrizio Montesi in the year 2006, which was a part of his studies at Bologna. Initially, the project started as a SOCK process calculus implementation, influenced by programming language of WS-BPEL and the CCS process calculus. With support for, tree-like data structures (XML, but with a syntax similar to C and Java), message types, typed session programming, integration with Java and JavaScript, mobility of code, containment of programs, and web programming Jolie extends SOCK. Jolie is like the replacement for XML-based

Orchestration language likewise WS-BPEL as it facilitates the orchestration of web services because it provides a C-like syntax to navigate XML-like data structures.

I. LITERATURE SURVEY

JOLIE language overview

[6]For developing orchestrator services, JOLIE offers a C-like syntax. The C-like syntax makes the language easy to understand for a programmer who is used to it. The JOLIE language has some simple introduction, with the exception and condition syntax similar to the C language.

Identifiers- An identifier is a name stored in the memory of the orchestrator which defines the location, a variable or a link and an operation. The identifier must be associated with the With the regular expression that follows [a-zA-Z]([0-9a-zA-Z])*. Some JOLIE statements enable the compiler to include a list of identifiers i.e. comma-separated identifiers (such as identifier11, identifier22, d, e, f).

Program structure- The following grammar represents the JOLIE software structure:

Program:

locations { Locations-definition∗ }

operations { Operations-declaration∗ }

variables { Variables-declaration }

links { Links-declaration }

definition∗

main { Process }

definition∗

definition:= define id { Process }

In which the star indicates a time repetition of zero or more.

Variables- Variables in Jolie are typeless. Implicit types are integers and strings that are supported. The non-terminal declaration of variables which contains set of identifiers describing memory variables that are shared.This definition describes: Variables-declaration: id list

Links- To internal concurrent operational synchronization, links are used.

Links-declaration: id list

Definitions- Definitions allows to define a procedure that can be called by another by using the call expression. Each definition combines a process with an identifier. The Method is conceptually a block of code composed of statements. As the body of a C function, the method is defined within a definition can be seen.

Main- The main block allows the process to be executed at the beginning of the execution of the stated program and it is equivalent to a C program's main function.

II. MICROSERVICES OVERVIEW

Microservices is a framework for the development of small dimensional distributed systems , dynamic service topology and loose coupling, derived from Service-Oriented Architecture. It is necessary to compose micro services to have other micro services in exchange. The language was originally created in the EU SENSORIA project the main formalization initiative for workflow and service composition languages, which gave rise to a range of reasoning frameworks for the composition of services. On the more functional side, Jolie comes with formally defined semantics, influenced by service-oriented computing standards such as WS-BPEL. Jolie's integration of theoretical and functional aspects allowed its use in correct-by-construction software analysis. By sharing messages, micro services can run together. Messages are structured as trees (a variant of the structures that can be found in XML or JSON). When messages are sent or received, correspondence is type-checked at runtime. . 3 For the creation of distributed applications in the cloud, micro services are particularly suitable. However, their dynamic nature calls for adequate strategies for their automated deployment. JRO (Jolie Redeployment Optimiser), a platform for the automated and optimized development of micro services written in the Jolie language. In standard SOAs, services are used as overlays to incorporate and organize autonomous information systems. This synchronization is accomplished by means of communications that function within standard protocols.[4]

What is Micro services Architecture?[5]

The architecture of micro services is a development methodology where fragments are turned into a collection of smaller services in a single program, where every program run in its own process and communicate with lightweight frameworks. Microservices are developed across business capabilities that can be independently distributed using an automatic deployment process. The architecture of micro services requires absolute necessities of management of these services developed into various programming languages and utilizes a range of data storage technologies.

Possible Languages for Micro services

A range of frameworks can be used to incorporate microservices, versions and tools. Java, Python, C++, Node JS and .Net.

Factors to Move into Micro services Architecture

- In a monolithic architecture, it's hard to grasp the complexities of a large application also sometimes, the code is difficult to handle.
- Sometimes applications do need thorough manual monitoring to understand the effects of manipulations and changes.
- The entire program has to be deployed again, just with a minor change.
- Heavy applications of monolithic architecture will delay the schedule time.

Benefits of Micro services

- Simpler Process Implementation –by the architecture of micro services, modern technologies and the implementation will be easier.
- Simple Modules-Since an application is divided into various sections, which is easy for developers to develop and manage it.
- Independent Scaling – In micro services, each module will scale independently by means of
    - X-axis scaling – Replicating for more RAM or Processor CPU
    - Z-axis scaling – By size using the sharing process
- DURS - In the architecture of micro services, each service can be DURS independently (Deployed, Updated, Replaced & Scaled)
- Unaffected - And a single module's loss would not impact the remaining portion of the application.

III. CONCLUSION

The motive of this article is to understand the importance of micro service and its implementation using the open source programming language Jolie as a language based approach. Micro services is an architectural style inhibiting from Service-Oriented Architectures (SOAs). The key concept is that applications are composed of small independent building blocks the Micro services interacting through a message passing. Jolie is a strict interpretation of theoretical orchestration process. The internal structure of the interpreter is particularly suitable for future improvements, with the possibility of developing an orchestrated system (On the same machine or on separate machines, by executing multiple instances of an interpreter).

IV. REFERENCES

1. https://en.wikipedia.org/wiki/Jolie_(programming_language) last accessed on 16 Aug. 2020.

2. https://www.Jolie-lang.org/about_Jolie.html last accessed on 16 Aug. 2020.

3. “Refinement types in Jolie” [cs.SE]22 Feb 2016.

4. “Self-Reconfiguring Microservices” M. Gabbrielli et al.

5.https://www.clariontech.com/blog/5-best-t

echnologies-to-build-microservices-architect ure last accessed on 30 Aug 2020.

6. Fabrizio Montesi1 , Claudio Guidi2 , Roberto Lucchi2 Gianluigi Zavattaro2 “JOLIE: a Java Orchestration Language Interpreter Engine” CoOrg 2006.