Chimera: Agnostic Language Component Based Framework using NodeJS and CLI

Go Frendi Gunawan
STIKI Malang
Malang, Indonesia
Email: frendi@stiki.ac.id

Mukhlis Amien
STIKI Malang
Malang, Indonesia
Email: amien@stiki.ac.id

Jozua Ferjanus Palandi
STIKI Malang
Malang, Indonesia
Email: jozuafp@stiki.ac.id

***Abstract*—Component Based Software Engineering (CBSE) is a branch of software engineering that emphasizes the separation of concerns with respect to the wide-ranging functionality available throughout a given software system. The main advantage of CBSE is separation of components. A single component will only focus on a single task or related collection of tasks. Allowing software developer to reuse the component for other use-cases. By using this approach, software developer doesn't need to deal with spaghetti code. Several approaches has been developed in order to achieve ideal CBSE. The earliest implementation was unix pipe and redirect, while the newer approach including CORBA, XML-RPC, and REST. Our framework, Chimera, was built on top of Node JS. Chimera allows developer to build pipe flow in a chain (a YAML formatted file) as well as defining global variables. Compared to unix named and unnamed pipe, this format is easier and more flexible. On the other hand, unlike XML-RPC, REST, and CORBA, chimera doesn't enforce users to use http protocol.**

Keywords—Chimera, Language Agnostic, Component-Based Software Engineering, CBSE, Node JS, CLI.

I. Introduction

Component based software development approach is based on the idea to develop software systems by selecting appropriate off-the shelf components and then to assemble them with a well- defined software architecture [1].

In order to implement component-based software engineering (CBSE), several different approaches has been performed by various companies and individuals. The very first implementation was Unix pipe mechanism [2]. Unix pipe allows engineer to pass output of a single program into as an input of another program. Since a lot of server is UNIX or linux based, this pipe mehanism availability is very high. Even DOS also provide similar mechanism [3]. Beside of it's high availability, at some point, UNIX pipe also support parallel processing through named-pipe mechanism. The named-pipe mechanism can be used to provide cheap parallel processing [4]. Although pipe mechanism provide high availability and capability, it has several limitations. For example, it needs external file as temporary container. The external file has to be deleted once the operation performed. Since this approach is not straight forward, some effort is needed in order to undestand or to build a named-pipe based computation. By it's nature, pipe mechanism itself has a drawback. Pipe mechanism assume a program's standard output turns into another program's standard output. Thus, in case of a program provide more than a single outputs or need to get several inputs, extra adjustment is needed so that the programs fit the mechanism.

Beside Unix pipe mechanism, another common approach to achieve CBSE has been introduced. The common approach usually involve CORBA, XML RPC, and RestFul architecture. Another interesting approach was named DEF - A programming language agnostic framework and execution environment for the parallel execution of library routines [5]. DEF focus on parallel processing by enabling shared memory and message passing. DEF needs several components, using JSON as data exchange format. Compared to CORBA, Matlab, and Parallel Fortran is better in term of parallelism and language agnosticism. CORBA for example, doesn't support matlab and octave [5]. However, DEF still depend on REST for communication. Consequently, in order to build DEF architecture, a web server is needed. Also, each component should follow certain rules in order to be able to communicate in DEF protocol.

Beaker Notebook [6] is also considered as an interesting approach of CBSE. The platform was developed by Two Sigma Open Source and mainly used for research use. Like DEF, beaker has a shared storage containing global variables. The global variables is accessible by any pieces of program in a certain notebook. However, extra works is needed so that a programming language is available in a notebook.

Based on previous approaches, it seems that Beaker Notebook and DEF is better in terms of language agnoticism and shared memory. However, Unix pipe is better in terms of availability and architecture independence.

In this paper, we propose a novel approach of CBSE. We name it Chimera. Chimera is an agnostic language component based framework using NodeJS and CLI. Chimera doesn't require specific architecture. Any program works in CLI will also works in Chimera. No adjustment effort is needed. Chimera also support sharing memory mechanism to share global variables. Thus, it also allows developer to define straight-forward process flow.

A. Subsection Heading Here

II. Conclusion

Appendix A
Proof of the First Zonklar Equation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra

sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] A. Kaur and K. S. Mann, "Component based software engineering," *International Journal of Computer Applications*, vol. 2, no. 1, pp. 105–108, 2010.

[2] M. D. McIlroy, J. Buxton, P. Naur, and B. Randell, "Mass-produced software components," in *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*, 1968, pp. 88–98.

[3] L. T. Two Sigma Open Source, "Dos 7 command," http://www.lagmonster.org/docs/DOS7/pipes.html, accessed: 2017-06-19.

[4] T. Conway, "Parallel processing on the cheap: Using unix pipes to run sas programs in parallel. sas users group international (sugi 28) proceedings. march 30–april 2, 2003. seattle, washington," 2003.

[5] T. Feilhauer and M. Sobotka, "Def-a programming language agnostic framework and execution environment for the parallel execution of library routines," *Journal of Cloud Computing*, vol. 5, no. 1, p. 20, 2016.

[6] B. Watson, "Beaker notebook," http://beakernotebook.com/index, accessed: 2017-05-30.