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 special protocol such as HTTP (except for distributed computing scenario). Nor it require the components to be aware that they works on top of the framework.**

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 approaches has beeen performed. The earliest attempt was UNIX pipe mechanism [2]. Pipe mechanism was not the only attempt to achieve CBSE. The more modern approaches including XML-RPC [3] and JSON-RPC [4]. Later, Object Management Group (OMG) introduced a new standard named CORBA (Common Object Request Broker Architecture) [5]. Another interesting approach was introduced by Two Sigma Open Source. Two Sigma created a platform known as Beaker Notebook [6]. Beaker Notebook is mainly used for research purpose. On 2016, Feilhauer and Sobotka introduce another platform called DEF [7].

Aside from Unix Pipe, all other mechanism require the components to be aware that they are part of the framework. This means that you cannot use old programs (e.g: cal and cowsay) as XML-RPC or CORBA component. At least additional layer and adjustment has to be built.

CORBA, XML-RPC, and JSON-RPC also needs HTTP protocol since they were designed for in client-server architecture. It imply that you need to build a web server in order to use the mechanisms. However, in any use case that only need a single computer, this is not ideal.

Considering the advantages and disadvantages of those early approaches, in this paper, we introduce a new CBSE framework named Chimera. This framework is much simpler since HTTP is only required for distributed computation. Chimera also use CLI mechanism that works in almost all OS and most programming language. The only dependency of Chimera are NodeJS and several NPM packages.

II. Previous Research

In this section we will have an indepth discussion about UNIX Pipe, XML-RPC, CORBA, and DEF.

A. UNIX Pipe

The very first implementation of CBSE was UNIX pipe mechanism [2]. UNIX pipe allows engineer to pass output of a single program 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 [8].

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 [9].

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. This approach is not straight forward, thus, some efforts is needed in order to to build a working named-pipe based computation.

By it's nature, pipe mechanism assume a program's standard output turns into another program's standard output. For simple use cases, this might be the best approach. However, at some point, when the program become more complicated, memory sharing and network access might be needed. Using a mere pipe mechanism to support those requirement is either hard or impossible.

B. XML-RPC, JSON-RPC, and SOAP

XML-RPC is a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the

Internet. XML-RPC using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned [3].

JSON-RPC is lightweight remote procedure call protocol similar to XML-RPC [4]. The main difference between XML-RPC and JSON-RPC is the data transfer format. In most cases, JSON is more lightweight compared to XML.

XML-RPC and JSON-RPC are heavily depend on HTTP for inter-process-communication protocol.

Beside UNIX pipe mechanism, another common approach to achieve CBSE has been introduced. The common approach usually involve CORBA, XML RPC, and RestFul architecture.

C. CORBA

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.

D. DEF

Another interesting approach was named DEF - A programming language agnostic framework and execution environment for the parallel execution of library routines [7]. 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 [7]. 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.

E. Beaker Notebook

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.

III. Our Approach

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.

IV. Conclusion

Chimera

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] Trac, "Json-rpc," http://json-rpc.org, accessed: 2017-05-30.

[4] I. UserLand Software, "Xml-rpc.com," http://xmlrpc.scripting.com, accessed: 2017-05-30.

[5] O. M. Group, "Corba," http://www.corba.org/, accessed: 2017-05-30.

[6] L. T. Two Sigma Open Source, "Beaker notebook," http://beakernotebook.com/index, accessed: 2017-05-30.

[7] 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.

[8] B. Watson, "Dos 7 command," http://www.lagmonster.org/docs/DOS7/pipes.html, accessed: 2017-06-19.

[9] 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.