applied sciences

MDPI

Article

A Vital Sign Analysis System Based on Algorithm Block Broker for Interoperability between Algorithm Development Tools

Moon-Il Joo [1] and Hee-Cheol Kim [1,2,*]

[1] Institute of Digital Anti-aging Healthcare, Inje University, Gimhae-si 50834, Korea; joomi@inje.ac.kr
[2] Department of Computer Engineering, Inje University, Gimhae-si 50834, Korea
* Correspondence: heeki@inje.ac.kr; Tel.: +82-55-320-3720

Abstract: With the recent development of artificial intelligence and data mining technology, various and intelligent vital sign analysis technologies have been developed. Vital sign analysis algorithms and technologies are primarily developed using MATLAB and open source technologies, such as Python and R. The analysis algorithms developed with such programming languages can only be employed and run in their own respective development environments and, hence, are unfortunately not considered as platform independent. In that respect, the interoperability between development tools is needed to ensure efficiency in terms of development time and efforts and reusability between analysis technologies and algorithms developed in different languages. This paper presents the development of a vital sign analysis system that ensures interoperability, which leads to one common environment connecting different development platforms. To maintain the interoperability between MATLAB and R programming, we designed and implemented the Algorithm Block Broker (AB Broker). AB Broker is composed of AB Adapter and AB Broker. Here, the AB Broker uses AB Adapter to request execution of analysis algorithms developed in different languages, such as MATLAB, R, and Python. It also searches and runs the algorithm, helping implement the requested analysis technique. The AB Broker-based vital sign analysis system enables the integrated management of analysis and data mining technologies developed in different languages. From a developer's point of view, therefore, it is convenient and efficient to develop techniques using existing different programming technologies.

Keywords: interoperability; data mining; vital sign; vital sign analysis

Citation: Joo, M.-I.; Kim, H.-C. A Vital Sign Analysis System Based on Algorithm Block Broker for Interoperability between Algorithm Development Tools. *Appl. Sci.* **2021**, *11*, 1913. https://doi.org/10.3390/app11041913

Academic Editor: Roger Narayan

Received: 8 January 2021
Accepted: 19 February 2021
Published: 22 February 2021

1. Introduction

The efficient handling and use of the considerable information and data currently available is recognized as an issue directly linked to the competitiveness of the industry in most fields of the Fourth Industrial Revolution [1]. Most industries take significant effort to handle the information and data overflow more efficiently and productively; the issue is directly related to the survival of the company in terms of business activities. Therefore, the importance of artificial intelligence technology to handle and analyze extensive information is increasing [2,3].

The application of artificial intelligence is also expanding in the medical sector; it has evolved from the mere creation of new information and knowledge through the use of data well, and has become more focused on developing and evaluating services [4]. In particular, in the field of medical and healthcare, services can be received anytime, anywhere other than in hospitals, unlike in the past, with the advent of wearable devices capable of measuring vital signs in daily life. Therefore, healthcare, which manages the health conditions of individuals, has shifted from being "disease treatment-oriented" to "prevention and management-oriented" [5,6].

Vital signs data have been increasing exponentially recently due to wearable devices capable of measuring various vital signs [7]. The increased information on vital signs has

the characteristics of big data, and various studies are being conducted to identify methods of disease prevention, management, diagnosis, and treatment using vital signs that have been converted to big data [8–10].

Vital sign analysis technology is important to provide healthcare services because of the development of artificial intelligence technology. Various artificial intelligence and data analysis technologies are distributed in an open source format using programming language with open source features, such as Python and R Programming [11,12]. Numerous vital sign analysis algorithm technologies are being developed because of the advancement of artificial intelligence technology.

In recent times, vital sign analysis algorithms have been developed using development tools, such as MATLAB, Python, and R Programming. These development tools are used, either by utilizing libraries or by being customized. However, the developed algorithms cannot be used in other programming languages because they are dependent on the programming language. Therefore, various studies have been conducted to obtain interoperability between different programming languages [13]. In particular, studies on interoperability between systems developed in different languages in a distributed environment have been conducted progressively [14]. However, studies have not been conducted on the interoperability between programming languages to develop artificial intelligence, which is important in the era of the Fourth Industrial Revolution. Therefore, interoperability studies are required to secure reusability between algorithms developed in different languages.

The algorithm block broker (AB Broker) was designed to secure interoperability between different languages; a vital sign analysis system, applying this AB Broker was developed in this paper. AB Broker calls and executes different algorithm functions between algorithm programming languages, such as MATLAB, Python, and R Programming. It has an algorithm block adapter (AB Adapter) and AB Broker. The AB Adapter designates the function to be called. The Broker will then obtain the execution result value by executing the function called from the AB Adapter. For example, R Programming calls a function developed in MATLAB through the AB Adapter. The Broker then executes the function developed in MATLAB and sends the result value to R Programming. Similarly, the algorithm source file itself developed in MATLAB and R Programming needs to be executed in the system to implement the vital sign analysis technology using AB Broker. Therefore, the functions are called using the Broker developed in Java to build an integrated environment.

The languages used for algorithm development have their own characteristics. Python has strengths in artificial intelligence by utilizing many open-source libraries. R programming has strengths in text mining from the beginning. In addition, MATLAB features an integrated development language, such as artificial intelligence and signal processing. However, MATLAB has traditionally been strong in signal processing and provides various libraries. It will be a cornerstone for developing better algorithms by taking advantage of the strengths of these development languages. Thus, the AB Broker of this paper has an operator that executes a function by passing arguments to execute a function and returns a result value. Such an operator automatically executes the function through the Java interface when the developer requests function execution through the AB Adapter. These operators have the advantage that developers can easily integrate functions or libraries developed in various languages. In addition, there is no interference between the development languages because the development languages utilize the results obtained by independently executing them. With the development of an interoperability-based system, research and development focused on simple vital sign analysis algorithms will shift to a high-level, service-oriented development that easily utilizes various algorithm sources.

2. Related Studies and Backgrounds

2.1. Related Studies

Data analysis tools, such as Python, MATLAB, R programming, and application development tools, such as C, C++, C#, and Java are being studied for interoperability with

development languages. Representatively, there is Common Object Broker Architecture (CORBA) [15].

CORBA is an interface protocol that allows software components written in different languages and running on different platforms to interact with each other. The method calling method between different languages uses Object Request Broker (ORB) [16]. In the early days, various studies were conducted by using ORB to call methods between Java and C++ languages. With the development of distributed computing with the improvement of high-performance PCs and networks that provide high bandwidth, research has been conducted to secure interoperability using CORBA in a distributed environment [17].

CORBA can easily integrate between different languages by calling methods. However, the calling method is a calling technique and combining methods developed by each language. In this way, different languages cannot connect to each other and execute methods of different languages. For example, while developing a method in Java, it is necessary in C++ language. You cannot use a method in Java.

A technology that allows different languages to connect and directly execute functions and commands of different languages has been studied. Representatively, there are rJava and Matlabcontrol.

rJava can execute R programming functions and commands by interlocking with R programming in Java [18]. These studies are being conducted to secure interoperability between java and R Programming. Interoperability research using rJava uses Java's graphical user interface (GUI) to overcome sophisticated graphic work, which is a drawback of R programming [19]. Representatively, this study graphically shows data analyzed using R Programming using JavaFX [20].

Matlabcontrol can execute MATLAB source code by connecting to MATLAB engine in Java. Representatively, research has been conducted on remote and virtual laboratories that remotely connect to the server where MATLAB is installed and execute MATLAB commands using Java and MATLAB interface, Matlabcontrol.

Research on the linkage between application development tools and data analysis tools has been conducted. However, until now, research on applying analysis tools developed in different languages is insufficient. Hence, research is needed to execute the desired algorithm by applying the source code developed with various analysis tools to the system.

2.2. Vital Sign Data Mining Techniques

Data mining is technology to explore and discover knowledge by modeling and searching for relations, patterns, and rules existing in the (big) data [21]. Study on algorithms to extract and mine useful and meaningful information from accumulated data are important, and it has been conducted in various areas, such as finance, medicine, education, and so on. In particular, data mining research for personal healthcare services becomes more and more important and actively carried out [22]. For example, IBM's Watson began to diagnose cancers at human level with artificial intelligence.

Vital signs that we tackle in the paper are largely divided into two types. The first type is the data by which its value is meaningful, e.g., blood pressure and body temperature, and the second one is time-series data to measure continuously over time, such as electrocardiogram (ECG), acceleration, respiration, and so on, which usually have waveforms. The second type is much more complex to analyze, requiring signal preprocessing. For example, ECG is the signal combined with five waves of P, Q, R, S, and T. One can measure useful information including exercise intensity, stress, and heart rates, and the like from ECG data. In particular, heart rate variability (HRV) extracted from ECG is also a valuable vital sign to get parameters, such as standard deviation of normal to normal interval (SDNN), low frequency (LF), high frequency (HF), and heart rate per minute. One can diagnose arrhythmia from HRV data. Recently acceleration data becomes crucial. Acceleration sensors are attached in many wearable devices and in smartphones, and so, acceleration data will play an important role as big data in a near future, in that they are collected everywhere and easily. Typical examples of the information analyzed from

acceleration data are: the number of steps, walking distance, activity patterns of sitting, standing, walking, and running.

3. AB Broker Design for Interoperability

The AB Broker proposed in this paper secures interoperability by developing algorithms developed in different environments in an integrated environment. The use of AB Broker is expected to improve the reusability of various algorithms and provide time for algorithm development.

In this paper, AB Broker was developed to secure interoperability between MATLAB, which is conventionally used for algorithm development among various algorithm development tools, such as Python, MATLAB, and R Programming, and open-source R Programming.

The application of AB Broker is not limited to only vital sign analysis. In this paper, the proposed approach is the first of its kind developed for vital sign analysis. However, this framework is applicable for other frameworks because it is a generalized framework and there are no restrictions. As shown in Figure 1, AB Broker consists of an algorithm search, algorithm specification agent, execution agent, Algorithm Block Registry (AB Registry), and AB Adapter.

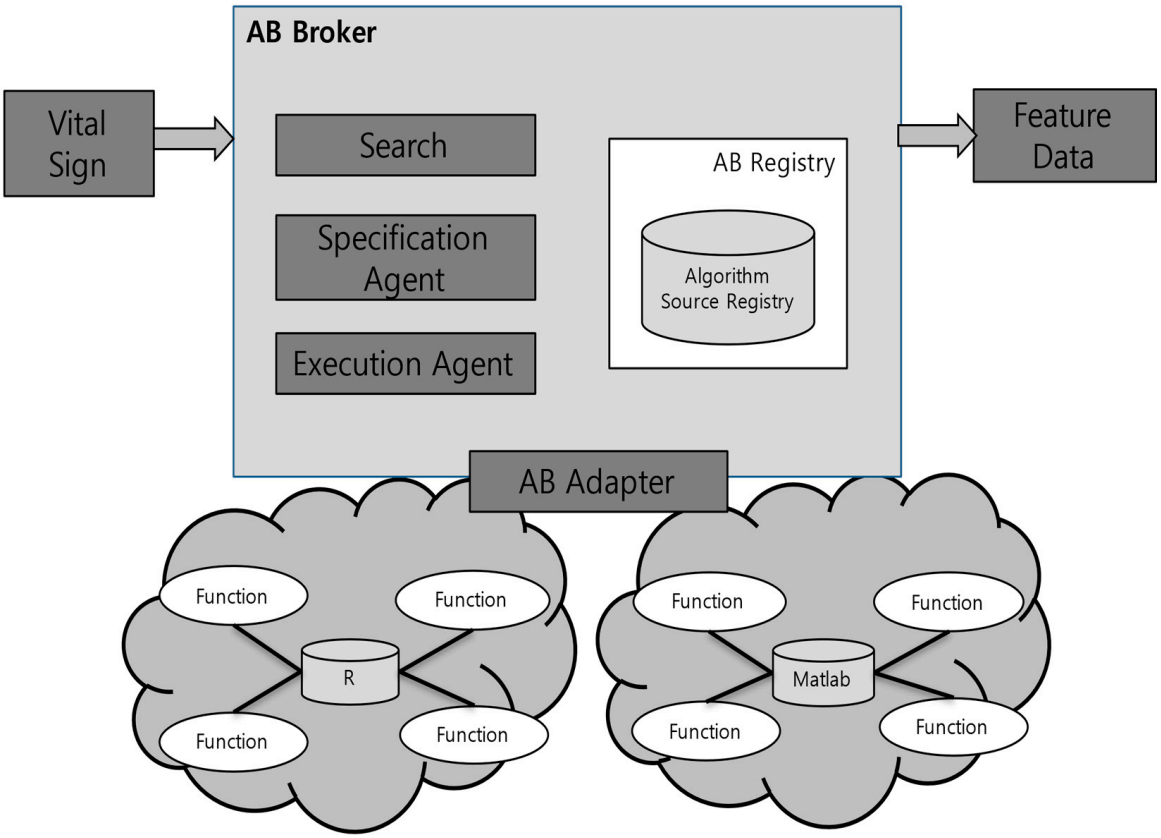

Figure 1. Algorithm Block (AB) broker diagram consisting of search, specification agent, execution agent, and AB adapter.

- Algorithm search searches for an algorithm to be executed.
- The algorithm specification agent checks the information on the algorithm to be executed and displays information on the type of development tool, the input value, and the output value required for execution.
- AB Registry is a repository for storing algorithm source files. It has the algorithm source file to execute the algorithm by searching for the algorithm that AB Broker requests to execute.

- AB Adapter has a service that stores information to call R programming functions in MATLAB and a service that saves information to call MATLAB functions from R programming. The information for executing functions is composed of parameters having the name of the function to be executed and an input value.
- Execution Agent executes the algorithm requested by AB Adapter. Algorithm execution has four steps. First, the presence of an algorithm is checked through algorithm search. Second, the algorithm function source is checked in the AB Registry. Third, the input and output values are checked according to the algorithm specification. Fourth, the algorithm is executed by inputting an input value to the algorithm function.

The algorithm search and algorithm specification agent analyze data using algorithm information. Algorithm information is stored in the algorithm information database shown in Figure 2. The algorithm details Table stores information on developer ID, algorithm file name, vital sign analysis algorithm explanation, vital sign type, date sent, and vital sign development tool. algorithm_inp_out stores information on input values, input parameters, and input values.

id	filename	explanation	vitalsings_type	reg_dt	tool_type
joo@wellness.com	average_pw.m	data average	ecg	2017-04-10	matlab
joo@wellness.com	dfilt2.m	dfilt filter	ecg	2017-04-10	matlab
joo@wellness.com	evalQRSDetection.m	ecg to hrv	ecg	2017-04-10	matlab
joo@wellness.com	filtspec.m	filtspec filter	ecg	2017-04-10	matlab
joo@wellness.com	hpassfilter.m	hpassfilter	ecg	2017-04-10	matlab
joo@wellness.com	lpassfilter.m	lpassfilter	ecg	2017-04-10	matlab
joo@wellness.com	standardDeviation.R	ecg to sdnn	ecg	2017-03-09	R

① algorithm_detailsTable

id	filename	parameter_order	inp_out	parameter_type	reg_dt	parameter_info
joo@wellness.com	evalQRSDetection.m	0	input	double array	2017-04-10	ecg data
joo@wellness.com	evalQRSDetection.m	1	input	double	2017-04-10	hz
joo@wellness.com	evalQRSDetection.m	0	output	double array	2017-04-10	R peak index
joo@wellness.com	evalQRSDetection.m	1	output	double array	2017-04-10	R peak value
joo@wellness.com	evalQRSDetection.m	2	output	double array	2017-04-10	R end index

② algorithm_inp_outTable

Figure 2. Algorithm specification table modeling: ① is a table defining algorithm information; ② is a table that defines the input and output values of the algorithm.

The structure of AB Adapter is shown in Figure 3. MATLAB Adapter and R programming Adapter request information to execute functions in MATLAB and R Programming.

The working principle of Algorithm Block Broker is shown in the Figure 4. AB Broker requests execution of functions developed in MATLAB or R Programming from AB Adapter. AB Adapter sends information for the execution of the algorithm to the Broker object. The Broker searches for a function in the AB Registry according to the function execution service request. The searched function checks information on the function of the algorithm specification agent. The accurate input and input value required for the execution of the function are checked according to this information. Then, for the function developed in R Programming, the R Programming function is executed through the Java/R Interface (JRI) in MATLAB and the result value is sent to MATLAB. In addition, for the function developed in MATLAB, the MATLAB function is executed through Matlabcontrol, and the result value is sent to R Programming.

```
function [value] = adapter(name, varargin)
setenv R_HOME /usr/lib/R
import wellness.service.Broker;
Broker.open(name);
for i=1:length(varargin)
      Broker.setVariable(varargin{i});
end
value=Broker.execute();
end
```

① Matlab Adapter

```
adapter <- function(name, param1, param2, param3, param4){
library(rJava)
.jaddClassPath("/home/hadoop/workspace/matlab_java/bin")
a<-.jnew("wellness.service.Broker")
.jcall(a,"V","open",name)
if(!missing(param1)) .jcall(a,"V","setVariable",.jarray(param1))
if(!missing(param2)) .jcall(a,"V","setVariable",.jarray(param2))
if(!missing(param3)) .jcall(a,"V","setVariable",.jarray(param3))
if(!missing(param4)) .jcall(a,"V","setVariable",.jarray(param4))
data<-.jcall(a,"[D","execute")
data
}
```

② R Programming Adapter

Figure 3. AB Adapter source code: ① is MATLAB adapter source code for requesting R programming execution in MATLAB; ② is the R programming adapter source code for requesting MATLAB execution in R programming.

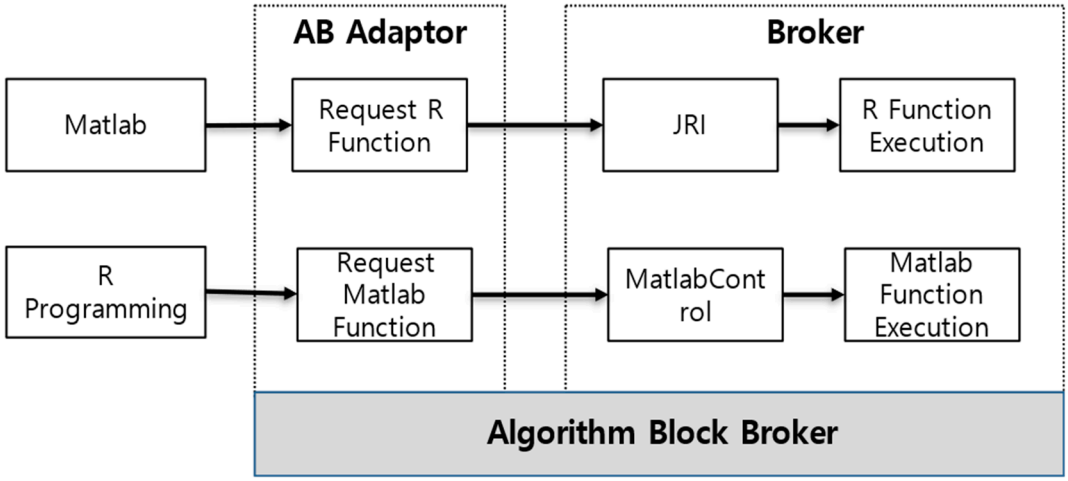

Figure 4. Scenario for calling and executing MATLAB and R programming in AB broker.

4. Vital Sign Analysis System Based on AB Broker

The proposed architecture is a model based on service oriented architecture (SOA) [23] in Figure 5, a type of web service. By SOA, the system searches services in the server that the developer wants and responds to the application, which is a suitable architecture for request/response concerning data mining techniques.

The vital sign analysis system architecture based on AB Broker has four scenarios:

- Requesting to execute vital sign analysis algorithm through Simple Object Access Protocol (SOAP) message.
- Retrieving vital sign data from vital sign storage.
- Executing vital sign analysis algorithm using AB Broker of execution engine.
- Saving the analyzed result in the data warehouse.

SOAP message provides a service to execute a vital sign analysis algorithm. The service is divided into a service that can search for a vital sign, a service that uploads a source file of a vital sign algorithm developed in MATLAB or R programming, and a service that extracts a vital sign feature value by executing a vital sign analysis algorithm.

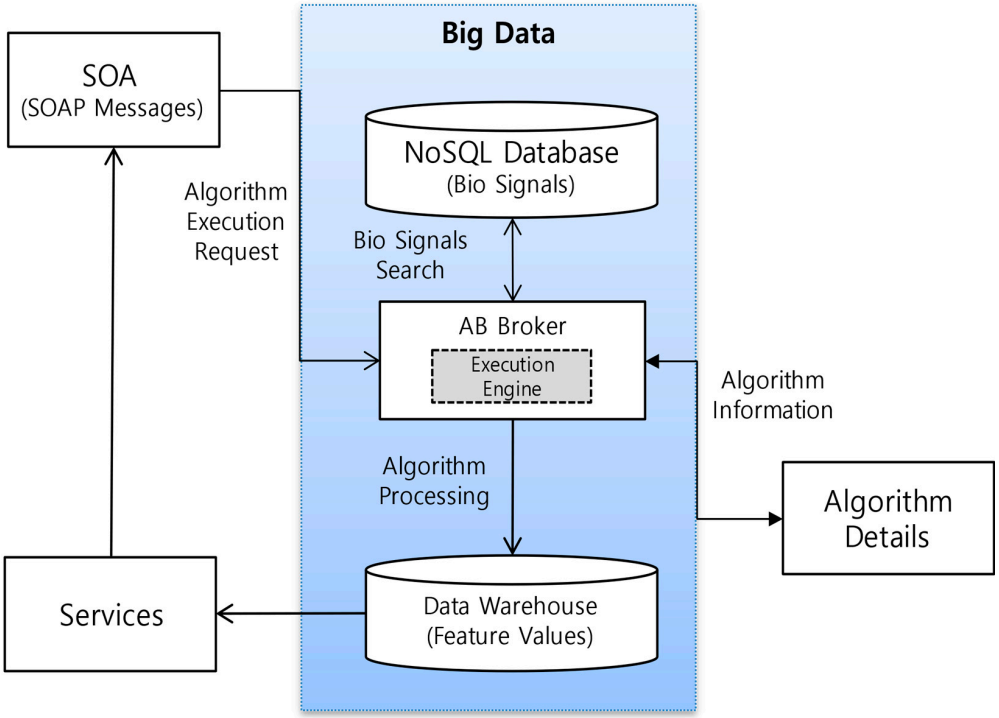

Figure 5. Vital sign analysis system architecture based on AB broker.

The NoSQL database stores various bio signals, in this case the vital signs. For such a design, a criterion for classifying vital signs is important. For example, signals such as electrocardiogram, electromyogram, respiration, etc., have continuous data. However, since the acceleration signal has continuous data of three axes (X-, Y-, Z-axis), the data of three axes must be stored at the same time. We designed a NoSQL-based vital sign storage that can collect and analyze a large amount of data, according to the characteristics of these vital sign.

The engine to execute vital sign analysis algorithm intends to apply the vital sign analysis algorithm developed by MATLAB and R programming to the health care service system. An interface for inputting an input value is required based on the specification of the vital sign analysis algorithm. In particular, an interface for retrieving a necessary input value from a vital sign storage is essential. Based on the input value, the vital sign execution engine performs pre-processing to convert the input value according to the development language to be executed, and then executes the vital sign analysis algorithm source.

The data warehouse stores the results from the execution engine. The data extracted by applying the vital signs analysis algorithm may be a great amount in a single column. Thus, the data are needed to save into the data warehouse for big data analysis.

5. Implementation and Application

In this paper, we have developed a vital sign analysis system based on AB Broker. This study executed the evalQRSDetection Function developed in MATLAB as shown in Figure 6. ① in Figure 6 is an AB Adapter used for executing the low-pass filter function developed in R programming, as shown in Figure 7. The evalQRSDetection function is an algorithm that analyzes the value of R Peak on the electrocardiogram (ECG). The evalQRSDetection function has two input values, electrocardiogram data, and sampling frequency. The two result values are the R peak index and R peak value.

```
function [maxIdx, maxVal] = evalQRSDetection(data, FS)
if nargin<2
   FS=100;
end
fs=FS;
fh=5;
maxIdx = [];
maxVal = [];
rawData = data;
dcRemData = rawData-mean(rawData);
lpData = adapter('lowpassfilter.R',dcRemData,60,200);
hpData = hpassfilter(lpData, fh, fs);
diffData = diff(hpData);
sqrData = diffData.*diffData;
window= ones(1,30);
integral= medfilt1(filter(window,1,sqrData),10);
delay = ceil(length(window)/2);
integralData = integral(delay:length(integral));
max_h=max(integralData);
thresh = 0.3;
peak_reg = integralData>(thresh*max_h);
sIndex = find(diff([0 peak_reg']))==1);
eIndex = find(diff([peak_reg' 0]))==-1);

for i=1:length(sIndex)
    [maxVal(i) maxIdx(i)] = max( hpData(sIndex(i):eIndex(i)) );
    maxIdx(i) = maxIdx(i)-1+sIndex(i);
end
if isempty(maxIdx)
    maxIdx = [1];
end
```

① AB Adapter

Figure 6. R peak extraction algorithm from electrocardiogram (ECG) developed by MATLAB: ① is the AB adapter to request the lowpassfilter. R function.

```
lowpassfilter <- function(data, lpfreq, sfreq){
lp_freq=lpfreq/sfreq
library(signal)
bf <- butter(2, lp_freq, type="low")
b1 <- filtfilt(bf, y+noise1)
b1
}
```

Figure 7. The lowpassfilter extraction function developed by R programming.

In this paper, you can check the source code shown below that calls the R Programming function using the AB Adapter in the MATLAB. Conversely, R Programming is a system that can call MATLAB function using the AB Adapter. Thus, the role of the adapter function is to keep away from the duplicated code when the interface of two languages is different, but the classes are same. The code shown below is an example, which shows that, originally, the codes are written in MATLAB; however, by using the adapter function, some of the code is written in R language, since both the classes are the same, although their interface is different. Grimmer et al. proposed framework, which was mainly developed for cross-language interoperability, to provide good runtime performance [24]. This shows that the proposed developed system shows interoperability between languages, which will be useful for the development of the better system for superior performance.

The vital sign analysis algorithm execution SOAP message and response SOAP message of Figure 5 are as shown in Figure 8.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ser="http://service.wellness">
  <soapenv:Header/>
  <soapenv:Body>
    <ser:getApplyAlgorithm>
      <!--Optional:-->
      <ser:id>joo@wellness.com</ser:id>
      <!--Optional:-->
      <ser:filename>evalQRSDetection.m</ser:filename>
      <!--Zero or more repetitions:-->
      <ser:input>1337 1359 1383 1409 1434 1457 1481 1505 1527 1545 1558 1564 1564
      1555 1542 1525 1503 1480 1457 1433 1406 1375 1339 1298 1251 1203 1156 1111
      1068 1025 986 949 916 887 863 843 827 814 803 795 789 783 779 780 783 791 801
      813 824 836 847 857 868 879 891 901 909 917 925 933 944 954 965 977 987 995
      1003 1009 1016 1023 1027 1029 1031 1036 1044 1051 1058 1064 1072 1082 1092
      1099 1104 1110 1114 1115 1115 1116 1114 1114 1115 1117 1118 1121 1124 1129
      1135 1141 1147 1153 1162 1174 1191 1224 1278 1355 1448 1505 1480 1407 1331
      </ser:input>
      <ser:input>250</ser:input>
      <!--Optional:-->
      <ser:toolType>matlab</ser:toolType>
    </ser:getApplyAlgorithm>
  </soapenv:Body>
</soapenv:Envelope>
```

① Algorithm Execution SOAP Message

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:getApplyAlgorithmResponse xmlns:ns="http://service.wellness" xmlns:ax27="http://report.wellness/xsd">
      <ns:return>15.0 91.0</ns:return>
      <ns:return>54.85248454132336 32.8066721884158</ns:return>
    </ns:getApplyAlgorithmResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

② Response SOAP Message

Figure 8. SOAP message of vital sign analysis algorithm execution: ① is a SOAP message to request algorithm execution; ② is the response SOAP message with the algorithm executed.

The request to execute the vital sign analysis algorithm is sent by inputting the ID that sends the vital sign analysis algorithm source to be executed, the vital sign analysis algorithm file name, input value, and the vital sign analysis development tool as shown in ① in Figure 8. The vital sign analysis algorithm is executed on the server and the result value is sent as shown in ② in Figure 8.

The evalQRSDetection Function execution UI and request result UI are as shown in Figure 9. The UI to execute the evalQRSDetection Function is as shown in ① in Figure 9. The execution result value is as shown in ② in Figure 9, where the first chart is the result value maxIdx in Figure 5 and the second chart is the result value of maxVal.

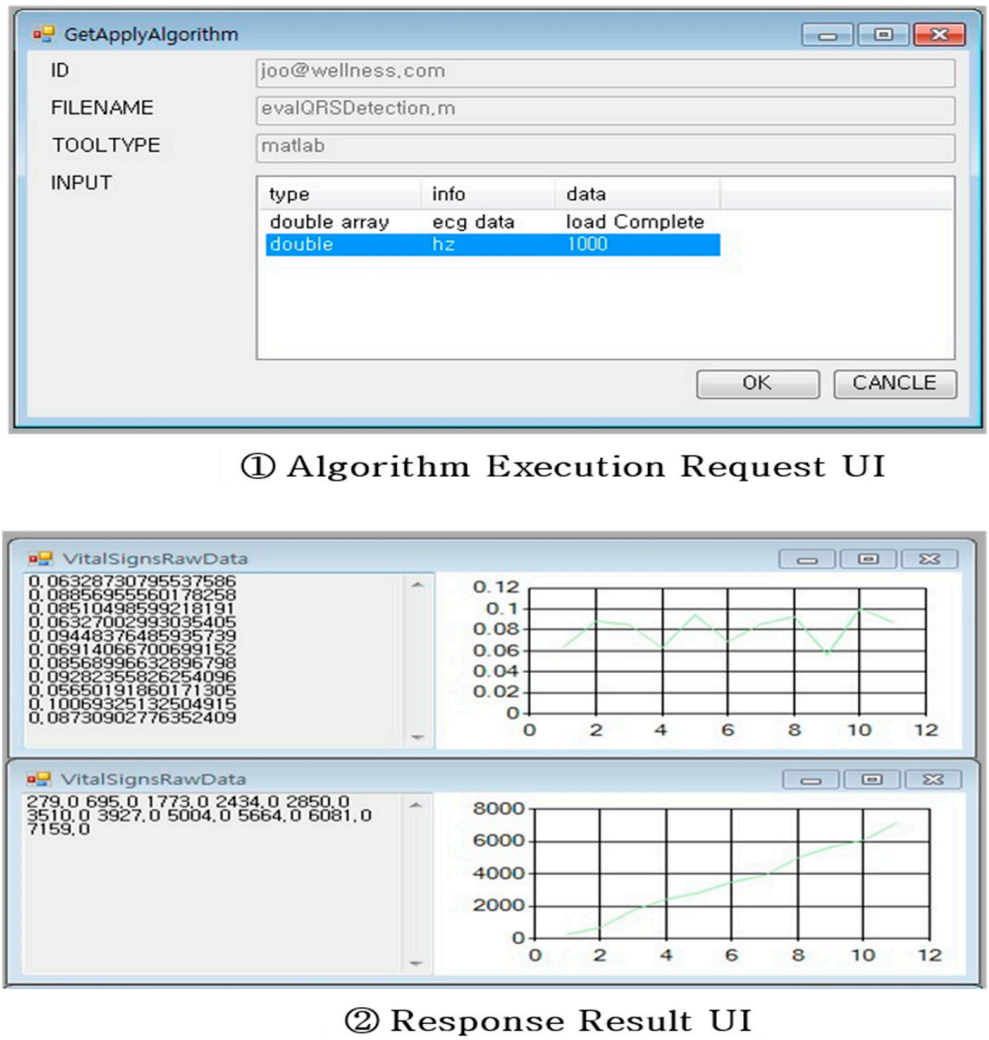

Figure 9. Implementation of vital sign system user interfaces: ① is a UI for requesting algorithm execution; ② is the response UI where the algorithm was executed.

6. Conclusions

AB Broker has been designed to secure interoperability between algorithms developed in different languages. The vital sign analysis system applying AB Broker provides an architecture that can easily apply various analysis algorithms to the system. In this paper, we implemented a vital sign analysis system using AB Broker-based MATLAB and R programming language.

Algorithm techniques developed in various development languages with the method presented in this paper can be easily used on one platform to secure interoperability. In addition, it has a software structure that can improve the reusability and convenience of analysis technology, which is expected to improve the quality of vital sign analysis technology and secure system competitiveness.

The limitation of this study is that in this paper, the proposed approach was tested only for the bio signals. However, the framework is developed by keeping in mind that it will be used as a generalized framework and can be used for other applications. In this paper, since our study is limited to bio signals, we only mentioned the bio signals.

Finally, in the future, AB Broker demands research on architecture that can be applied to systems that require various artificial intelligence technologies by adding Python and other development languages and analyzing vital signs.

Author Contributions: Conceptualization, M.-I.J. and H.-C.K.; methodology, H.-C.K.; software, M.-I.J.; validation, M.-I.J. and H.-C.K.; formal analysis, H.-C.K.; investigation, M.-I.J.; resources, M.-I.J.; data curation, M.-I.J.; writing—original draft preparation, M.-I.J.; writing—review and editing, H.-C.K.; visualization, M.-I.J.; supervision, H.-C.K.; project administration, H.-C.K.; funding acquisition, H.-C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, through the Education Program for Creative and industrial Convergence. (Grant Number N0000717).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Basic Science Research Program through the National Research Foundation of Korea (NRF), supported by the Ministry of Science, ICT & Future Planning (NRF2017R1D1A3B04032905).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brondoni, S.M.; Zaninotto, E. Ouverture de 'The 4th Industrial Revolution. Business Model Innovation & Global Competition'. *Symph. Emerg. Issues Manag.* **2018**, *2*, 1–7.
2. Pan, Y. Heading toward Artificial Intelligence 2.0. *Engineering* **2016**, *2*, 409–413. [CrossRef]
3. Huang, M.H.; Rust, R.T. Artificial intelligence in service. *J. Serv. Res.* **2018**, *21*, 155–172. [CrossRef]
4. Miller, D.D.; Brown, E.W. Artificial intelligence in medical practice: The question to the answer? *Am. J. Med.* **2018**, *131*, 129–133. [CrossRef] [PubMed]
5. Lee, K.Y.; Kim, J. Artificial intelligence technology trends and IBM Watson references in the medical field. *Korean Med. Educ. Rev.* **2016**, *18*, 51–57. [CrossRef]
6. Kim, T.W.; Park, K.H.; Yi, S.H.; Kim, H.C. A big data framework for u-healthcare systems utilizing vital signs. In Proceedings of the 2014 International Symposium on Computer, Consumer and Control, Taichung, Taiwan, 10–12 June 2014; pp. 494–497.
7. Gaskin, J.; Jenkins, J.; Meservy, T.; Steffen, J.; Payne, K. Using wearable devices for non-invasive, inexpensive physiological data collection. In Proceedings of the 50th Hawaii International Conference on System Sciences, Honolulu, HI, USA, 4–7 January 2017.
8. Elgendi, M. Less is more in biosignal analysis: Compressed data could open the door to faster and better diagnosis. *Diseases* **2018**, *6*, 18. [CrossRef] [PubMed]
9. Ta, V.D.; Liu, C.M.; Nkabinde, G.W. Big data stream computing in healthcare real-time analytics. In Proceedings of the 2016 IEEE International Conference on Cloud Computing and Big Data Analysis, Chengdu, China, 5–7 July 2016; pp. 37–42.
10. Chen, C.P.; Zhang, C.Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Inf. Sci.* **2014**, *75*, 314–347. [CrossRef]
11. Milne, D.; Witten, I.H. An open-source toolkit for mining Wikipedia. *Artif. Intell.* **2013**, *194*, 222–239. [CrossRef]
12. Rao, A.R.; Clarke, D. A fully integrated open-source toolkit for mining healthcare big-data: Architecture and applications. In Proceedings of the 2016 IEEE International Conference on Healthcare Informatics, Chicago, IL, USA, 4–7 October 2016; pp. 255–261.
13. Grimmer, M.; Seaton, C.; Schatz, R.; Würthinger, T.; Mössenböck, H. High-performance cross-language interoperability in a multi-language runtime. In Proceedings of the 11th Symposium on Dynamic Languages, New York, NY, USA, 21 October 2015; pp. 78–90.
14. Blair, G.S.; Bennaceur, A.; Georgantas, N.; Grace, P.; Issarny, V.; Nundloll, V.; Paolucci, M. The role of ontologies in emergent middleware: Supporting interoperability in complex distributed systems. In Proceedings of the ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Lisbon, Portugal, 12 December 2011; pp. 410–430.
15. Zinky, J.A.; Bakken, D.E.; Schantz, R.E. Architectural support for quality of service for CORBA objects. *Theory Pract. Object Syst.* **1997**, *3*, 55–73. [CrossRef]
16. Schmidt, D.C.; Levine, D.L.; Mungee, S. The design of the TAO real-time object request broker. *Comput. Commun.* **1998**, *21*, 294–324. [CrossRef]
17. Vinoski, S. CORBA: Integrating diverse applications within distributed heterogeneous environments. *IEEE Commun. Mag.* **1997**, *35*, 46–55. [CrossRef]
18. Urbanek, S. rJava: Low-level R to Java interface. Available online: http://cran.rediris.es/web/packages/rJava/ (accessed on 11 September 2020).
19. Furtuna, T.F.; Vinte, C. Integrating R and Java for Enhancing Interactivity of Algorithmic Data Analysis Software Solutions. *Rom. Stat. Rev.* **2016**, *64*, 29–41.
20. Bistak, P. Remote laboratory server based on Java Matlab interface. In Proceedings of the 2011 14th International Conference on Interactive Collaborative Learning, Piestany, Slovakia, 21–23 September 2011; pp. 344–347.

21. Jain, N.; Srivastava, V. Data mining techniques: A survey paper. *Int. J. Res. Eng. Technol.* **2013**, *2*, 116–119.
22. Birnbaum, E.B.D. Application of data mining techniques to healthcare data. *Infect. Control Hosp. Epidemiol.* **2004**, *25*, 690–695.
23. Sprott, D.; Wilkes, L. Understanding service-oriented architecture. *Archit. J.* **2004**, *1*, 10–17.
24. Grimmer, M.; Schatz, R.; Seaton, C.; Würthinger, T.; Luján, M.; Mössenböck, H. Cross-language interoperability in a multi-language runtime. *ACM Trans. Program. Lang. Syst.* **2018**, *40*, 1–43. [CrossRef]