THE MIDDLEWARE ARCHITECTURE DESIGN FOR GATHERING THE HETEROGENEOUS DATA IN BIG DATA

Charoenporn Bouyam

School of Informatics, Walialak University,
222 Thaiburee, Thasala, Nakornsrithammarat, THAILAND, 80160
charoenporn.bo@mail.wu.ac.th

Chanankorn Jandaeng

Informatic Innovation Center of Excellence (IICE),
School of Informatics, Walialak University,
222 Thaiburee, Thasala, Nakornsrithammarat, THAILAND, 80160
chatchanan.ja@mail.wu.ac.th

Abstract
The data varieties are the important problem of the variety of the system in an organization. The difference in database design and implementation affects the data integration. More than 30 systems are explored in different times and technology. Although the system and data are distributed, the dashboard generation is unique. It is hard to access web applications via new devices and new behaviors. The middleware is the intermediate node that acts as a proxy to collect, combine, and re-format to be the general format. The paper presents the software architecture and workflow of middleware is to collect data from the various system in the organization. The middleware is evaluated with 7 performance metrics by 7 experts. The result shows that the average score of overall metrics is 2.47 of 3.00. This middleware is easy to program and implement with a microservice model to increase scalability, availability and extensibility.

***Keywords*: middleware design: heterogeneous data: data collection: big data.**

1. Introductions

Web applications and web sites are important tools that support internal operations for the Thai government for a long time. Walailak University is a Thai government organization that implemented many web-based systems such as human resources management, finance management, e-document and central student registration. All systems are designed and implemented as web-based applications. All applications are accessed via the internet web site. Web applications increase the flexibility, comfortability, and productivity of workflow since 1998.

Whereas the evolution of smartphone technology is rapid. Users access the internet via the varieties of mobile devices and screen sizes such as mobile phone and tablet. In addition, the software system of mobile devices is transformed from embedded software to a multitasking operating system comparable to a personal computer. The internet speed is developed from the analogue signal in the second generation (2G) to the high-speed digital signal in the fifth generation (5G).

For the academic's administration, they are many reports is to support decision support. The reports are collected that bring to analysis, synthesis the new knowledge, and present in the chart or descriptive report. Because each data sources are distributed in many and various platform and format. The data are growth in 10 Terabyte. It is hard to get the report within a few minutes for each report. These are the basis of big data's characteristics based on 3Vs. model (Volume, Variety, Velocity)(Sivarajah et al., 2017).

This paper addresses the problem in a variety of platform and data format. Each system is managed by the internal department and established in a different timeline the bring to the difference in web programming technology and database technology. A report that concern with the data from many subsystems is hard to process.

The intermediate nodes are required. These node act as the middleware is to gather the variety of data format form the variety of host. The middleware is the intermediate node to receive the request message via web API. Then, it gathers data from any subsystem.

This paper proposed the software architecture design of middleware for gathering data in the organization that prepare for the other subsystem. The author described the functional module, data flow between client, middleware including exchanged document and security issue. Then, the architecture is evaluated by the experts with 7 performance metrics.

2. Related Works

The middleware for Internet of Thins (IoT) have some characters like our work. It collects data form the various sensors or subsystem to store at server. Thus, other application as the client gather data in order to analyze to process in the next step.

Cyril Cecchinel (Cecchinel et al., 2014) defined the software architecture to collect the sensor-based data. The architecture gets the physical sensor data to store on cloud. This architecture supports the IoT to be the data collection for the bigdata project. The big data project is the Data as a Service (DaS) for the cloud dimension that provided the data via service API.

Zhi-yong Bai (Bai et al., 2016) proposed the design and implementation the multi-interface gateway for wireless sensor network. The gateway gathers sensing data via a various of wireless communication module. The core function of middleware is the reading data of wireless module via serial port and extract data from network frame.

Eirini Liotou (Liotou et al., 2017) presented the middleware architecture for quality of experience-driven service management. This middleware is based on microservice-based architecture. The middleware integrates the various modern system architecture technology such as software-defined network, big data, network functions virtualization, and QoE-based module and function.

Michele (Ciavotta et al., 2017) designed and implemented the middleware based on microservice paradigms to increase the security and privacy of the industry. The proposed idea was implemented within a frame of the MAYA European project to serve the IoT in the factory.

Daibin Wang (Wang et al., 2017)proposed the middleware to transparent the permission manager on Android OS. The permission manager is sensitive data and service. They propose a secure usable and transparent OS-level. They implemented to evaluate the CPU and memory usage, overhead, energy consumption. The middleware is to be secure against any permission leaks under the minimal impact on the usability of running application, transparent to app users and developer, and low in overall performance overhead.

Yasser Mesmoudi (Mesmoudi et al., 2020) proposed the middleware based on service-oriented architecture that apply in Internet of Thing. This middleware collected sensing data from the heterogeneous sensor hardware using REST API and heterogenous network such as WiFi, Bluetooth, and Zigbee. The collecting data is stored in gateway that support in various platform. This middleware is evaluated by implementing Raspberry Pi to monitor temperature and humidity.

Tiaco Cointo (Coito et al., 2020) proposed the middleware for intelligent automation to support the industrial. The middleware acts as the gateway that connects the field devices, database, and decision support system (DSS) in a real-time situation. Moreover, cloud services and database warehousing are used to address some of the challenges that are handled using fog computing and a multi-workload database. They demonstrated the middleware in the pharmaceutical industry, providing interoperability and real-time reaction

3. The Software Architecture Design

The middleware design starts with the user requirements. The requirement brings to gather and blend the information to be a single and standard format. The software architecture base on layer architecture. This section describes the user requirement analysis, the functions of each layer, and data flow in software architecture.

3.1. *The User Requirement Analysis*

The case study is the organization of government that was established in 1998. All business processes are controlled with web applications. Most of the web application is implemented by the outsource company via the auction. Base on government regulation, the committee cannot specify the company to implement the new application. Users draft and distribute the term of use (TOR). After that, the winner of the auction implements the application based on the TOR. User cannot specify web technology or software architecture. 21 years ago, there are 43 web applications (systems) still active that categorized into 4 groups: human resources management, academics, education services, and operation support. There were implemented with web 2.0 technology or two-tiers architecture whereas some web-based services are implemented with modern technology or n-tiers architecture. A few services are implemented with 3 tiers architecture. The single sign-on authentication system was not implemented. It results in user's databases are distributed and are not consistency. All services are web applications that access via desktop. They do not support multiple screen sizes especially mobile or tablet. Moreover, they do not provide the data export function or any API for directly connecting.

3.2. *The Software Architecture Design*

From the user requirements, we proposed the software architecture for middleware. The middleware installed in the intermediate system to transform the web API to be the querying data, calling function, connecting the existing web API. On the hand, this system collects data from the existing system and transform the result into the JSON document that is the standard format in this system. The proposed architecture is based on layer architecture that is consist of 4 layers: presentation, business, data, and cross-layer and is shown in Figure 1.

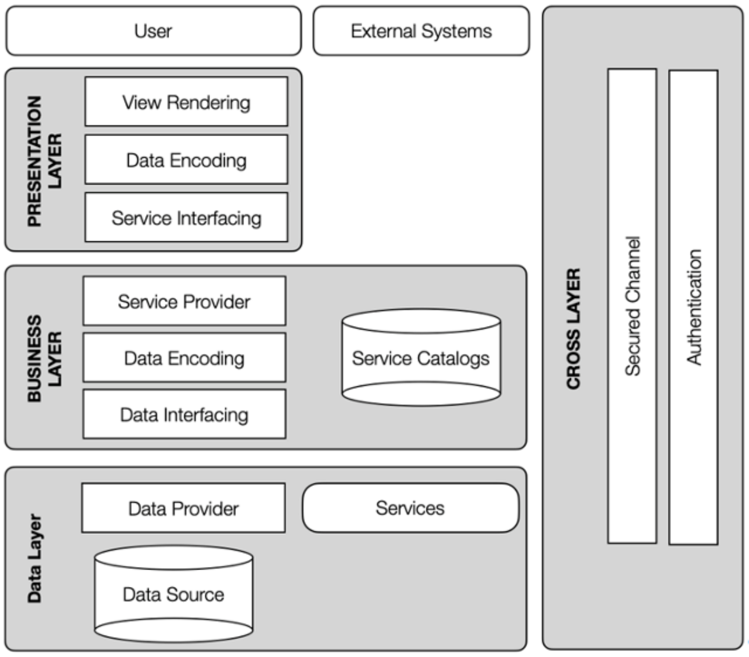

Fig 1. The Software Architecture of Middleware

Presentation layer: This layer is the top layer that interfaces users and external system. Users access the system via mobile/web application while the external system connects via service API. The client module transparent a big and complex system to be a unique application. This module is implemented with any technology such as web application with responsive web design, or native/cross-platform mobile application. Our design is composed of three parts: view rendering, data encoding, and service interfacing. The view rendering module is to acquire input and forward it to the service request module. Then, the message is transformed into parameters of the API message and sent to the middleware via the service interfacing part. The request message is processed in the business layer that acts as the middleware side, then the reply message is sent in JSON format. The JSON format is rendered in the view rendering module. Some services need to authenticate. The middleware replies message to acquire the username and password or authority in any service. Thus, our design proposes the authentication part that is the data collection of usernames and passwords and acts as the single sign-on system. Moreover, the external system is the other application that request to connect and gather data from our system. They request the data via API with parameters and process the data like the mobile/web application. In order to increase comfortability, this research proposed a new user interface like a web application supporting the responsive web design and single sign-on authentication. All registration services are accessed via these applications. The middleware acts as the software agent to query or call web API from varieties of services. The middleware is to transform the reply message to JSON format that ready to process with the JSON engine in the web browser.

Business layer: The middleware receives the request message via web API from the client module. The services provider module processes the message. If any request needs the security token or username and password, the message will be replied to ask for authorization. Then, the request message with username and password (if they need) is transformed into the web API that depends on the services categories. The web API may be the HTTP link, the HTTP is routed to call web service, or the generated HTTP request directly connect to the services. The service replies to the data to the middleware in the original format. Examples of reply format are HTTP document, the record set from the database, or JSON object from web API. However, the middleware transforms all reply to messages into the JSON format that brings to easy to render in the client module. The mechanism is divided into two categories: the statics page and the dynamics page. The static page is a web page implemented with basic web technology. The contents do not change at any time. It may be the static HTML file including the server-side script generated files such as PHP, JSP, or ASP, etc. Their file is easy to access via link. The static page is the news page, announcement, regulation, or any static information to distribute in the organization. The dynamics page is a web page that the content depended on the parameters. The parameters are submitted via the GET and POST method in HTTP protocol. It is implemented with server-side script languages such as PHP, Java, or .Net framework. There are 2 technologies that are applied in web applications: 2.0 and 3.0 technology. The server-side scripts are divided into 2 technologies: the procedural based and framework-based web page. The procedural web page represents that there is a function of web page files such as insert, update, delete, or query within a webpage. Whereas the framework base web page is clear-cut different. Each function is represented with a URL route. The syntax of the query string in the GET method is adapted because of the security issue. The critical parameters are sent via the POST method over the HTTPS protocol.

Data layer: This is the data provider and service of the system. The data providers are represented with service. All services are implemented with 2-tiers or 3-tiers architecture. The 3-tiers architecture consists of 3 parts: user interface, middleware, and database. User input data via the user interface. After that, the data is transferred to the database via middleware. The advantage of the 3-tier architecture is that the user interface depends on user devices such as desktop applications, web pages, or mobile applications. The 3-tier architecture is easy to implement in the proposed system by registering the API module and specify the parameters of the API. However, there are 20% of services are based on 3 tiers architecture. Most of the services are implemented with 2-tiers architecture and it does not support responsive web design. It is more difficult to adjust. Because business logic and user interface are designed and implement together. Therefore, it is necessary to rewrite the API part in order to join the proposed system. The migration process of the 2-tiers architecture application is providing the query command to the data broker. Then, the data broker directly accesses the database as the middleware of the 3-tiers architecture. Service Categories.

Cross layer: This layer is the supporting module that consists of three modules: secured channel, communication, and authentication. The message passing between the system in the proposed system is encrypted to protect the data disclosure by an unauthorized user. The selected protocols are HTTPS. However, there is some service still communicate via HTTP. Most services need to authenticate with username and password method. There is a centralized authentication server implemented with LDAP. Moreover, some system authenticated with themselves username. The authentication module is the cache of username and password to transparent the authentication mechanism.

3.3. *The Software Architecture of Middleware*

The software architecture is transformed and presented in 3-tiers architecture: client, middleware, and services. The client is the fronted module that is implemented with any technologies such as desktop application, native/cross-platform application, or web-based/web mobile application. The middleware acts as the intermediate nodes are to receive the request from the client and transform it to be web API and transform the result from the services to the general form. The services are the existing services that are implemented with the varieties of technology and platform. The software architecture is shown in Figure 2.

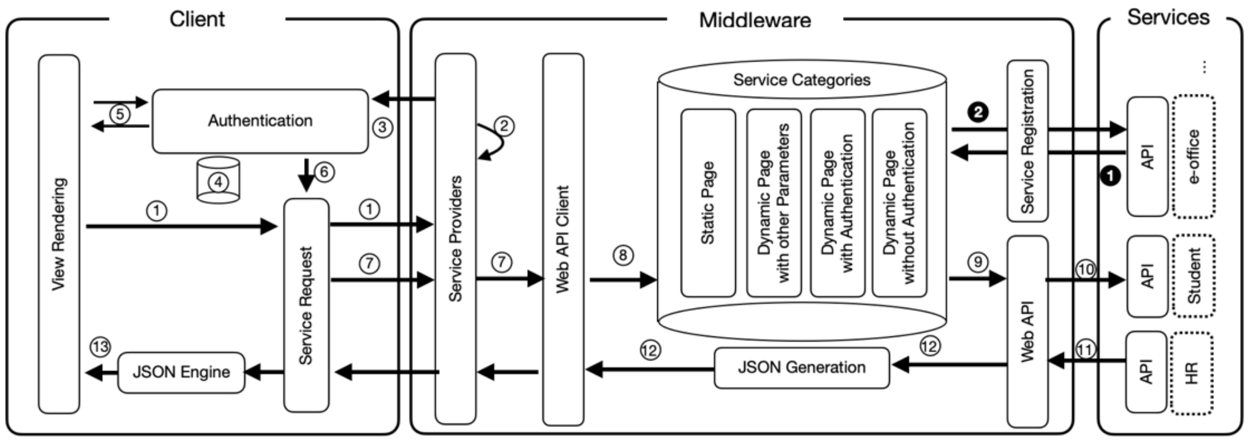

Fig 2. The Software Architecture of Middleware

The processes of the architecture are explained in the steps below:

Step 1: The user accesses via the user interface that renders via the view rendering module such as mobile application. The request messages are sent by the HTTP protocol.

Step 2: The service provider in the middleware module verifies the authorization such as token or password. The request message is forwarded to the authentication process in the client module when there is no authorized token.

Step 3-6: The authentication process lookups the authentication information for each request message in the small database or cache storage. On the other hand, it will request more username and password or authentication information from the user when the token is expired or there do not store in the cache. If the authentication token ready, the request message attached with the token is sent to middleware again.

Step 6-9: The authorized requests message connects the service via a web API client that gathers from service categories. The services categories consist of services in term of a static page, a dynamic page with the parameter, a dynamic page with authentication/without authentication mechanism. The service categories transform the general request message to be the specific request message (depend on service) and call the service via web API.

Step 10-13: The data from the service are replied to via Web API and transform to be JSON message. The JSON message is forwarded via web API client and service provider of middleware. Finally, a JSON message embedded in the HTTP response is sent to the web client. The JSON message is processed and rendered with a web engine or another client.

All services are registered in service categories via the service registration module. The information for registration is the function name and its parameters, authentication method, service category, etc.

4. Evaluation

The evaluation framework shown in Figure 3 consists of 3 parts: middleware, research method, and metrics. The middleware is the main issue of this paper, and it is explained in previous topic. This paper evaluated the middleware with quantitively approach. The questionnaire is distributed to the experts as participant.

4.1. *Population and Sample*

There are 7 software developers to evaluation the propose method. The experts in software development consists of senior programmers, 3 junior programmers, and 2 system analysts. The experiences distribute between 2 and 10 years in the software development domain. All developer understands the n-tier web architectures and expertise in programming language/framework at least 2 frameworks.

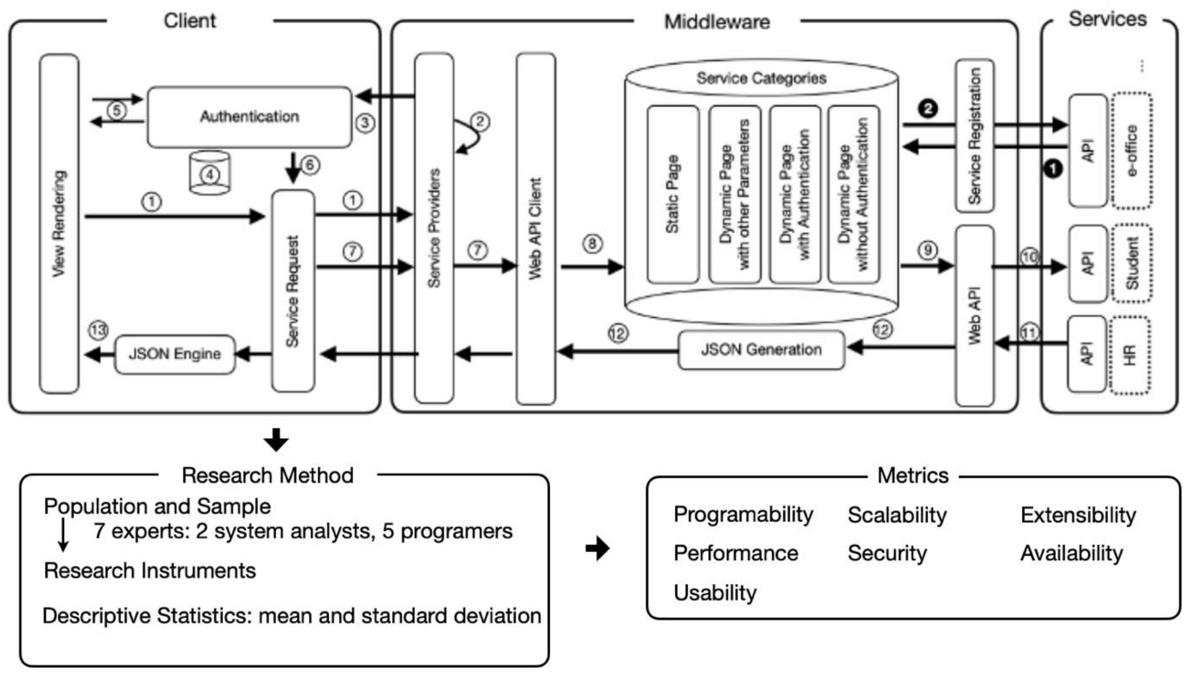

Fig 3. Evaluation Framework

This research selects the expertise with purposive sampling technique (Palinkas et al., 2015) because there are all system analysts and software developers in the department.

4.2. *Research Instruments*

The research instrument is composed of three parts: background of expertise, performance metrics, and open question is to acquire more comments. The performance metrics are composed of programmability, scalability, extensibility, security, availability, performance, and usability (Mesmoudi et al., 2020). Moreover, the suitability, compatibility, maintainability, and portability are the additional performance metrics (Ismail Kh & Hamarash, 2020).

The quality scale is divided to 3 levels: low, middle, and high. The description of performance metrics and scale of qualities are shown in Table 1.

The questionnaire is validated by five experts in information technology and statistics. The Index of Item-Objective Congruence (IOC) (Turner & Carlson, 2003) is used to find the instrument validity. The experts consist of one statistics expert and two software developer. The IOC is used to evaluate the instrument based on score between -1 to +1. The items have scores lower than 0.5 were revised. On the other hand, the items have scores more than equal 0.5 were reserved. The IOC of research instrument is 0.80 and standard deviation is 0.44. We conclude that this research instrument suite for the performance evaluation of our middleware.

4.3. *Result and Analysis*

The raw data are analyzed with descriptive statistics: average and standard deviation. The overall results are shown in Figure 4 with the axis representing the performance metrics.

The spider web chart in Figure 4 consists of 7 performance metrics. Each metric is composed of 3 levels: low, medium, and high. The average value of each metrics shows under the title. The experts give the highest score for the programmability and the low score for the performance.

The overall performance of middleware evaluated by experts is 2.47 + 0.26. The highest performance metrics is programmability (2.86) while the lowest is performance (2.14). The details of performance are discussed below:

Because most experts are the expertise in computer programming, they can develop web application both 2-tiers and 3-tiers architecture. They comment that the proposed middleware is the simple concept and easy to implement. On other hand, they need to know more details and precise for the service categories technique.

This middleware based on the 3-tiers architecture, so it has advantages in scalability, availability, and extensibility. In addition, the microservice model is an implementation method that provides the three metrics. An outstanding feature of microservices is scalability. Whereas availability can be managed at the service level (Liang & Lan, 2021) and implemented on the microservice system such as docker swarm or Kubernetes. Kubernetes is the automated deployment, scaling, and self-recovery when any service fails. Thus, it supports the availability. The middleware is designed for the distributed system that results in our approach support the extensibility.

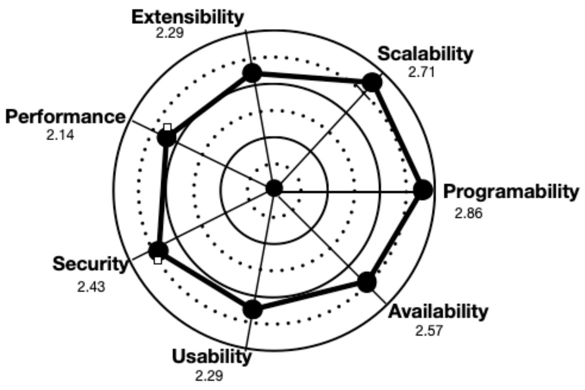

Fig 4. Overall evaluation of middleware architecture

For the usability issue, the message exchanging of the middleware is based on universal resource identity (URI). Data is encoded in readable JSON format. Most parameters are easy to understand. However, it depends on the implementation step and the developer.

This middleware is secured by two approaches: secured communication channel and secured programming. The secure socket layer protocol or HTTPS protocol is selected that depended on the architecture. The SSL is selected for the transport layer while the HTTPS protocol is used in the application layer. Whereas secure programming depends on the experience of development. The developer carefully selects and program with the secure instruction and avoid the flaw function. Moreover, the secured web framework is the other choice.

Metrics		**Performance**
Description		Concern the response time and round-trip time are estimated by experts
Question		How long do you estimate the response time or round-trip time within a middleware?
Quality Scale	High	Soft-real time that based on user requirement
	Medium	Maybe soft-real time. The worst case does not cause and affect the service.
	Low	Miss real time and some process affect the service.
Metrics		**Scalability**
Description		Concern the ability in handle increased volume of requests
Question		With 1,000 concurrent requests, how effectively do you think a proposed middleware can handle them?
Quality Scale	High	More than 95% of request can be handled
	Medium	More than 80% of request can be handled
	Low	More than 50% of request can be handled
Metrics		**Availability**
Description		If any part of middleware fails, cause an interruption of service. Thus, it concerns the failure possibility and fast recovery time
Question		Is the proposed system fault-tolerance and recoverable in time (because of middleware only)?
Quality Scale	High	The service uptime more than 95% and fast recovery
	Medium	The service uptime more than 80% and fast recovery
	Low	The service uptime more than 50% and fast recovery
Metrics		**Usability**
Description		Concern the goals of ease of use. Does this design is to eliminate the complexity of data gathering and provide the integrated information.
Question		With the external developer perspective, this middleware is easy to call or View implementation.
Quality Scale	High	It easy to connect and use the web API without consult.
	Medium	It easy to connect and use the web API with online consult.
	Low	The calling function may be hard to understand. The programmer needs to be trained.
Metrics		**Security**
Description		Consider the security in communication between client/middleware/services
Question		How the middleware design aware in the security issue both the function in middleware to the external system support.
Quality Scale	High	Fully support the secure communication or most security issue are supported by operating system or programming technique.
	Medium	Partial support the secure communication or some security issues are not aware.
	Low	The proposed middleware does not aware the security issues or it hard to implement the security function in middleware.
Metrics		**Programmability**
Description		Consider the ability in programming technique or technology support including the experience of programmer
Question		How the developer needs to learn the new technology/framework/algorithm is to implement the proposed middleware
Quality Scale	High	The traditional/ trend programming language or framework is enough.
	Medium	Some function or module need to be trained the new programming or framework
	Low	Most functions/module need to train the new programming language/framework.
Metrics		**Extensibility**
Description		concern in level of effort such as implementing, configuration and deployment task. Moreover, it includes the vertical system scalability such as adding compute node or distributed computing.
Question		How the proposed middleware design supports the implement as the distributed system or high-level architecture.
Quality Scale	High	There is no effect
	Medium	Some modules need to be changed before implement in distributed system.
	Low	The proposed middleware does not support distributed system.

Table 2. The Quaestiones and Quality scale

5. Conclusions

To eliminate the problem of data gathering for big data processing, this paper proposes the middleware for collecting data in the intranet of the organization. The middleware is evaluated with the seven experts via questionnaire and interview for information in deep. The middleware is ready to implement with 3-tiers web architecture. In another word, the experts concerned about the security issue. However, the security issue is eliminated by a secured communication channel with HTTPS or SSL protocol. And, secured programming with any programming technique. The other performance metrics are supported by the microservice model implemented with the container concept.

References

[1] Bai, Z. Y., Kuo, C. H., & Wang, T. C. (2016). Design and implementation of an IoT multi-interface gateway for establishing a digital art interactive system. International Journal of Ad Hoc and Ubiquitous Computing, 21(3), 157–170. https://doi.org/10.1504/IJAHUC.2016.075376

[2] Cecchinel, C., Jimenez, M., Mosser, S., & Riveill, M. (2014). An Architecture to Support the Collection of Big Data in the Internet of Things. International Workshop on Ubiquitous Mobile Cloud, 442–449. https://doi.org/10.1109/services.2014.83

[3] Ciavotta, M., Alge, M., Menato, S., Rovere, D., & Pedrazzoli, P. (2017). A Microservice-based Middleware for the Digital Factory. Procedia Manufacturing, 11, 931–938. https://doi.org/10.1016/j.promfg.2017.07.197

[4] Coito, T., Martins, M. S. E., Viegas, J. L., Firme, B., Figueiredo, J., Vieira, S. M., & Sousa, J. M. C. (2020). A Middleware Platform for Intelligent Automation: An Industrial Prototype Implementation. Computers in Industry, 123, 103329. https://doi.org/10.1016/j.compind.2020.103329

[5] Ismail Kh, T., & Hamarash, I. I. (2020). Model-Based Quality Assessment of Internet of Things Software Applications: A Systematic Mapping Study. International Journal of Interactive Mobile Technologies (IJIM), 14(09), 128–152. https://doi.org/10.3991/ijim.v14i09.13431

[6] Liang, Y., & Lan, Y. (2021). TCLBM: A task chain-based load balancing algorithm for microservices. Tsinghua Science and Technology, 26(3), 251–258. https://doi.org/10.26599/TST.2019.9010032

[7] Liotou, E., Marotta, A., Pomante, L., & Ramantas, K. (2017). A middleware architecture for QoE provisioning in Mobile Networks. IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, 2017-June. https://doi.org/10.1109/CAMAD.2017.8031640

[8] Mesmoudi, Y., Lamnaour, M., El, Y., & Tahiri, A. (2020). A Middleware based on Service Oriented Architecture for Heterogeneity Issues within the Internet of Things (MSOAH-IoT). Journal of King Saud University - Computer and Information Sciences, 32(10), 1108–1116. https://doi.org/10.1016/j.jksuci.2018.11.011

[9] Palinkas, L. A., Horwitz, S. M., Green, C. A., Wisdom, J. P., Duan, N., & Hoagwood, K. (2015). Purposeful Sampling for Qualitative Data Collection and Analysis in Mixed Method Implementation Research. Administration and Policy in Mental Health and Mental Health Services Research, 42(5), 533–544. https://doi.org/10.1007/s10488-013-0528-y

[10] Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. Journal of Business Research, 70, 263–286. https://doi.org/10.1016/j.jbusres.2016.08.001

[11] Turner, R. C., & Carlson, L. (2003). Indexes of Item-Objective Congruence for Multidimensional Items. International Journal of Testing, 3(2), 163–171. https://doi.org/10.1207/s15327574ijt0302_5

[12] Wang, D., Yao, H., Li, Y., Jin, H., Zou, D., & Deng, R. H. (2017). A Secure, Usable, and Transparent Middleware for Permission Managers on Android. IEEE Transactions on Dependable and Secure Computing, 14(4), 350–362. https://doi.org/10.1109/TDSC.2015.2479613

Authors Profile

Charoenporn Bouyam is a lecturer at the School of Informatics in Walailak University. He gained bachelor's degree in Management Information System and master's degree in Management Information Technology from Walailak University, Thailand. His research interests are focused on medical information system, application development, and brain computer interaction.

Chanankorn Jandaeng is a lecturer at the School of Informatics in Walailak University. He completed the B. Sc and M. Sc. in Computer Science. Moreover, he received the Ph. D. degree in Computer Engineering from Prince of Songkla University, Thailand since 2012. Areas of interest are computer networks security, intelligence technology, and algorithm and programming technique in resource constrained devices.