A MODEL AND IMPLEMENTATION OF ACADEMIC DATA INTEGRATION IN NEAR-REAL TIME USING MESSAGE-ORIENTED MIDDLEWARE TO SUPPORT ANALYSIS OF STUDENT PERFORMANCE IN THE INFORMATION TECHNOLOGY DEPARTMENT OF POLITEKNIK CALTEX RIAU

Yuni[1)], Ardianto Wibowo[2)], and Muhammad Arif Fadhly Ridha[3)]
[1)]Information System Study Program, Politeknik Caltex Riau
[2)]Computer Engineering Study Program, Politeknik Caltex Riau
[3)]Informatics Engineering Study Program, Politeknik Caltex Riau
e-mail: yuni15si@mahasiswa.pcr.ac.id[1)], ardie@pcr.ac.id[2)], fadhly@pcr.ac.id[3)]

ABSTRAK

Pemanfaatan data telah secara efektif berkontribusi pada pertumbuhan institusi dengan memberikan wawasan untuk tujuan manajerial. Di Jurusan Teknologi Informasi (TI) Politeknik Caltex Riau, sistem informasi dibangun secara terpisah, sehingga sulit bagi kepala program studi untuk menganalisis kinerja akademik. Untuk keperluan analitis, ada sistem bisnis inteligensia yang dikembangkan untuk menyediakan setiap kepala program studi di Departemen TI pengetahuan tentang departemen mereka. Sayangnya, sistem bisnis inteligensia ini belum dilengkapi dengan integrasi data. Untuk mengatasi masalah ini, penelitian ini mengusulkan 2 model integrasi data real-time waktu akademik yang berbeda yang didokumentasikan menggunakan Enterprise Integration Pattern dan membuat tolok ukur implementasi untuk mendapatkan model integrasi data terbaik. Model menggunakan Message-Oriented Middleware, sebuah teknologi yang memungkinkan komunikasi asynchronous antara berbagai aplikasi. Penelitian ini menggunakan WSO2 ESB sebagai alat MOM dalam Service-Oriented Architecture (SOA) yang menggunakan library NuSOAP untuk membantu menghasilkan layanan web WSDL dan akan menggunakan pendekatan Integrasi Aplikasi Perusahaan. Pengujian dilakukan berdasarkan aspek ISO 9126: fungsionalitas, efisiensi, dan keandalan. Berdasarkan hasil pengujian, dapat disimpulkan bahwa kedua model integrasi memenuhi aspek fungsionalitas dan keandalan, tetapi pola ke-2 lebih efisien karena memiliki saluran pesan dan penyimpanan yang berbeda untuk setiap dimensi dan tabel fakta.

***Kata Kunci**: Integrasi Data, Message-Oriented Middleware, Near-Real Time ETL, Service Oriented Architecture.*

ABSTRACT

Data utilization has effectively contributed for institutions growth by providing insights for managerial purposes. In Information Technology (IT) Department of Politeknik Caltex Riau, information systems were built separately, makes it hard for the head of study program to analyze academic performance. For analytical purposes, there's a business intelligence developed to equip each head of study programs in IT Department with knowledge about their department. Unfortunately, the business intelligence hasn't considered with data integration. To solve this problem, this research proposes 2 different academic near-real time data integration model that are documented using Enterprise Integration Pattern and benchmarks the implementation to obtain best data integration model. The models use Message-Oriented Middleware, a technology that enables asynchronous communication between diverse applications. This research uses WSO2 ESB as the MOM tools in Service-Oriented Architecture (SOA) that use NuSOAP library for helping generating web service WSDL and use Enterprise Application Integration approach. The testing is conducted based on ISO 9126 aspects: functionality, efficiency, and reliability. Based on the testing results, it can be concluded that both integration models fulfill the functionality and reliability aspects, but the 2nd pattern is more efficient because it distinct message channel and store for each dimension and fact table.

***Keywords**: Data Integration, Message-Oriented Middleware, Near-Real Time ETL, Service Oriented Architecture.*

I. INTRODUCTION

TODAY effective use of data is important in industry competition. Industry decides faster response by utilizing historical and operational data. Business intelligence has been proven to effectively help the industry grows by providing better insights of data, trends, thus providing necessary information for managerial purposes.

In Politeknik Caltex Riau, especially in Information Technology (IT) Department, insight of students' data in each study program is not available. For analyzing academic performance, head of study programs gather data from number of systems i.e. academic, admissions, and research & community services to support the decision-making process. In addition, the decision-making process greatly relies on person who currently serves as the head of study programs. If a new head of study program is selected and transition process is not being held properly, then analytical process is possibly be unsynchronized. Whereas, the academic analytical process is critical to deal with existing problems. In this case, existing study focuses to create business intelligence system [1]. However, existing information systems are built individually. The existing business intelligence system has not considered with data integration.

This research is aimed for data integration model that support business intelligence requirements. The integration connects three objects: academic system, admission system, and data warehouse. Data integration model must ensure the data to be complete, accurate, and trusted. In addition, it has to be timely manner.
The proposed study designs Academic Data Integration Model in Near-Real Time using Message-Oriented Middleware. Two integration models benchmarked to find the best integration model that fits the requirement. Message-Oriented Middleware (MOM) is implemented technology that helps to integrate applications to communicate in an asynchronous way. The integration receives a message from a first software system, transforms at least a portion of the message into a generic format. Furthermore, it transforms the received message into a format that fits the intended recipient. In this research, the system utilizes WSO2 ESB as a tool to transform the received message into accepted XML format. The research also implements Service-Oriented Architecture (SOA). With the proposed architecture model, the implementation eventually produces fast and reliable data integration for the business intelligence.

II. LITERATURE REVIEW

A. Previous Work

Various research in the area of real-time ETL processes and Data Integration Model have been published. Author in [2] proposed a related research that implemented data integration based on the need of data integration between central pharmacy and branch pharmacy using web service in each pharmacy and utilized WSO2ESB as the middleware and using NuSOAP as the web service library.

Another research by [3] designed near real-time data warehouse integration model for accommodating integration and access to a big amount of data in a retail business using Service Oriented Architecture that used Change Data Capture (CDC). The web service also used NuSOAP as the web service library.

Other related research is [4], where this research designed and implemented data integration between several systems for mapping lecturers' works, researches, and community services. The integration implemented using RESTful web service from CodeIgniter Rest Server Library. The result is a system to show all the integrated data for efficient assessment of lecturers' work and research.

Further related research is proposed by [5]. The research used Windows Communication Foundation (WCF) as the integration framework and used MSMQ, as the middleware to send asynchronous message from one service endpoint to another. This research utilized Business Activity Monitoring (BAM) to capture data from SOAP envelope in WCF request or response to minimize latency.

This research focuses on designing and implementing near-real time data integration model between information systems and data warehouse using Message-Oriented Middleware, NuSOAP as the SOAP library, where the research is implemented in Enterprise Application Integration approach.

B. Theoretical Basis

1) Data Integration Approaches

There are several approaches that successfully applied data integration technique. The most popular approaches are such as following:

1. Extract Transform Load
   This approach is commonly used for feeding large volume of data from operational databases into data warehouses. Mostly, ETL is utilized in data warehouse.
2. Enterprise Information Integration
   Enterprise Information Integration (EII) emerged to be a solution for integrating multiple sources without having to load the data into a central warehouse [6]. EII allows this by focusing on query management.
3. Enterprise Application Integration
   Enterprise Application Integration is an integration framework that allows unrestricted sharing of information between two or more enterprise applications [7]. This framework consists of technologies and services, like middleware, to enable the exchange of message and transactions within the enterprise.

2) *Near-Real Time ETL*

Near-Real Time ETL is a mechanism for data warehousing to ensure the data in the data warehouse can be timely, which is beneficial for highly used system [8]. Contrary to traditional ETL, in Near-Real Time ETL, the data loading is performed continuously. By using Near – Real ETL, it is beneficial for the end user as the data source is more up-to-date and it can provide better data insight.

Near – Real ETL can be implemented in various way, e.g.:

a. Change Data Capture (CDC)
CDC is applied aligned with stream processor. To integrate both of data, stream processor and CDC are connected to a message queue. CDC stores any data change in a log. The data source reads CDC to update the data.
b. Real Time Data Cache (RTDC)
By using RTDC, the temporary real time data is stored, as the cache can be updated periodically to move its content to the data warehouse.
c. Middleware
Middleware can bridge communication between systems and provide mechanism to store data for each update.

3) *Service – Oriented Architecture*

Service – Oriented Architecture is a type of architecture that is service- oriented where services are provided to the other components through a communication protocol over a network [9]. Service in SOA is a group of functions or procedures, or process that respond to clients' requests. The service-orientation concept approaches the problem by dividing it to smaller services to finish specified problems. SOA makes best practices for the development in the field of Enterprise Application Integration (EAI) because of the approach allows reusability, interoperability, distributed deployment, and composability within a system.

4) *Web Services*

Web services are platform and language independent standard defining protocols in XML based that used internet to interact with other applications [2]. This technology adopts open standard such as SOAP, HTTP, and XML that allows web service to be implemented by different vendors without concerning the language or the platform. The key benefit of web services is they allow the system connected to each other without the need of massive reengineering.

Web service offers some benefits and flexibilities such as:

a. Interoperability
Web service's characteristic, platform and language independent, enables data exchange and communication between computers with different operation systems. Web service can also be accessed by diverse programming language, such as PHP, JSP, Java, VB.Net. Web service can also be accessed by mobile device.
b. Bridging Communication to Database
Traditionally, application needs database driver to be able to communicate with a database. Web service can act as a bridge between application and database so that application doesn't need to use database driver.
c. Facilitate Data Exchange
Because web service can be accessed by diverse platform and language program, it makes data exchange easier.

5) *Message – Oriented Architecture*

Message-Oriented Middleware is a software or hardware infrastructure that provides messaging capabilities between distributed systems on the basis of the asynchronous interaction mode. For the deployments, MOM acts as intermediary for exchanging the messages between the senders and the recipients [10]. A primary benefit of using MOM is the loose coupling between the systems, facilitate the data integration without having to adapt the source and the system to each other.

MOM provides built-in services for its implementation to maximize the utilization in enterprise scale. According to [10], there are several services available:

a. Message Filtering, where it allows the receiver to select the received messages based on certain criteria.
b. Transactions, providing ability to group tasks together.
c. Guaranteed Message Delivery, by saving all the messages in a nonvolatile storage, and resend the message when the consumer doesn't receive the message in a certain amount of time.
d. Message Formats, allows the user to use several formats available, such as Text, Object, Stream, HashMaps, Streaming Multimedia.

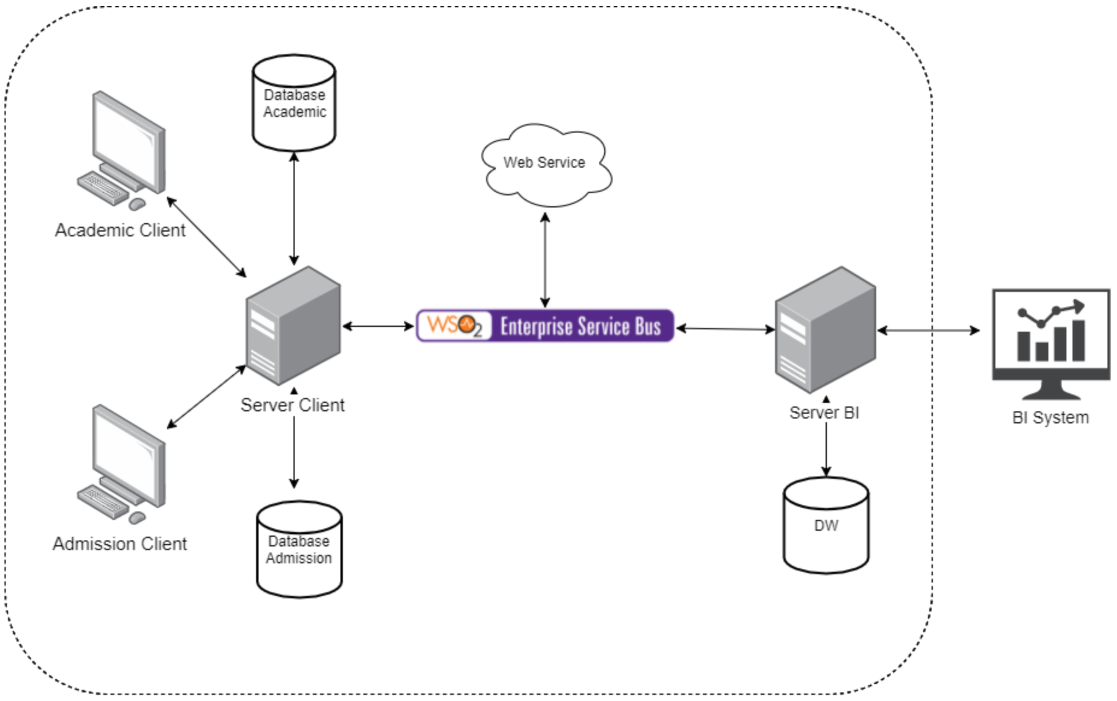


Fig. 1. System architecture.

e. Load Balancing, spreading the workload over the server with the lightest load.
f. Clustering, distributing an application over multiple servers and maintaining a single entity seamlessly.

6) Enterprise Integration Pattern

Enterprise Integration Pattern is a pattern that portrays how different applications integrate, to ensure that separate applications work together to produce a unified set of functionalities [11]. Enterprise Integration Pattern shows how the process of data delivery through a message exchanges between applications.

III. RESEARCH DESIGN

A. System Architecture

The proposed model designs the messaging procedure between the academic system, admission system, WSO2ESB, and Data Warehouse as shown in Figure 1.

Instead of receiving and updating data on a scheduled basis, the data warehouse is updated per row as an XML message. This message is sent from WSO2 ESB to the data warehouse after each update takes place. The client systems (academic system and admission system) sends a request to a certain proxy service within WSO2 ESB that implements the same method being consumed by the request, where this request may contain parameters that is included in the XML message. Then, WSO2 ESB send the returned value from the method to the data warehouse.

B. Messaging Pattern

1) First Messaging Pattern

As shown in Figure 2, for the first pattern, there’s only one channel that is used as the in-sequence. For the first pattern, all the input is processed the same: the data are sent to Router Service and are sent to each message store as described in the message. After being stored, the fact_table message and dimension_table message are processed by a Scheduled Forwarding Message Processor, to ensure the data are inserted successfully to the Data Warehouse.

2) Second Messaging Pattern

As shown in Figure 3, for the second pattern, the channel is differentiated for each dimension and fact table. The content-based router determines the type of message and the right message storage by implementing external PHP service. The storing mechanism is the same as the first pattern, using Scheduled Forwarding Message Processor and external service.

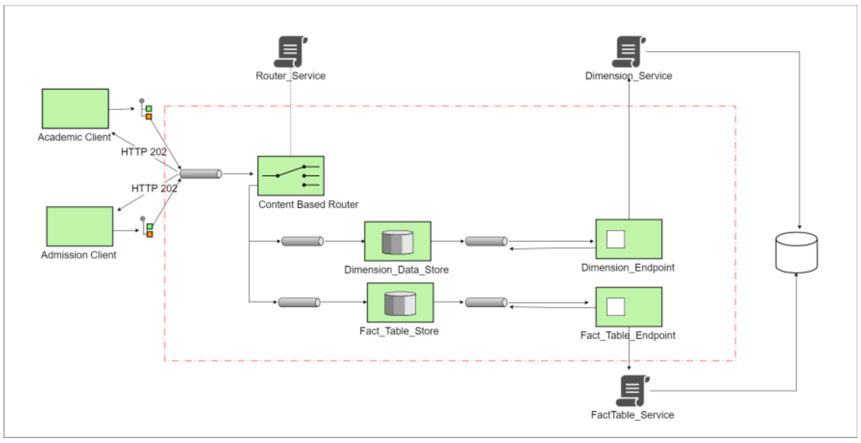


Fig. 2. First messaging pattern.

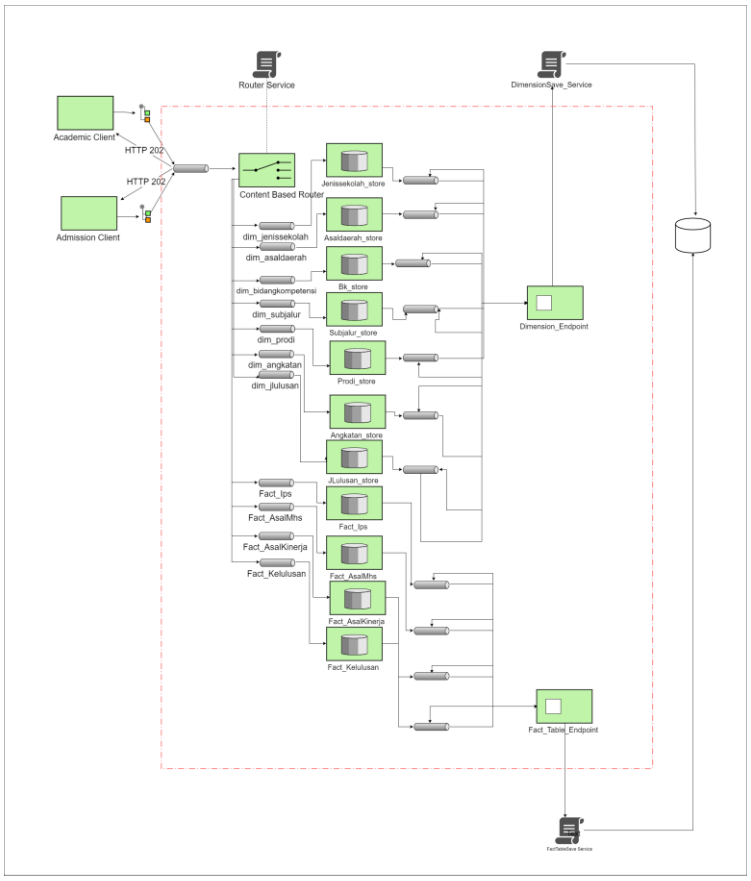


Fig. 3. Second messaging pattern.

C. *Data Illustration*

For this research, the data transformation is illustrated differently for each pattern. The illustration is based on the “prodi” data transformation to “dim_prodi” data.

1) *First Pattern Data Illustration*

In the first pattern, first the client sends request by calling the service’s function to insert “prodi” data (shown in Figure 4). The result is returned by service, transforming the data into the fitting format of data warehouse, including the identifier as shown in Figure 5. The result is returned to the middleware and is processed by content-based router and inserted to the defined message store as shown in Figure 6. After stored, the message is forwarded to external service and is inserted to data warehouse.

2) Second Pattern Data Illustration

On the second pattern, each dimension and fact table have different channel, so there’s no need to use identifier. First client send request to service. The difference is the return message from the service which only contain “daftar_dim_prodi” as shown in Figure 7. The message store is differentiated for each dimension, so the message is stored to “prodi_store”, as shown in Figure 8.

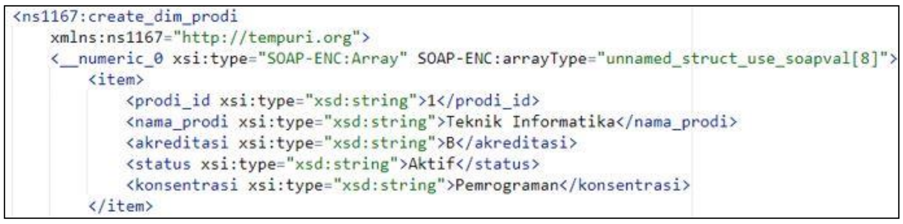
After stored, the message is forwarded to external service and is inserted to the data warehouse.

IV. IMPLEMENTATION AND ANALYSIS

A. Integration Implementation

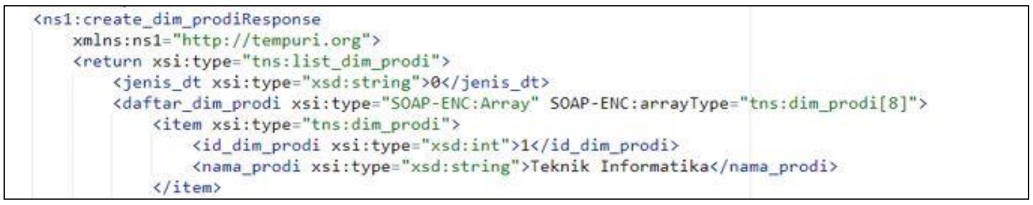
For the integration process, the trigger is put on a button in each prototype pages. Once the button is clicked, the client sends a request filled with operational data to the middleware and the data are processed. The trigger button is shown in Figure 9. The middleware logs the incoming message as shown in the Figure 10.

After that, the message channel reads the response and determine which service to send the message. The service only accept the message that fit the WSDL description as shown in Figure 11.



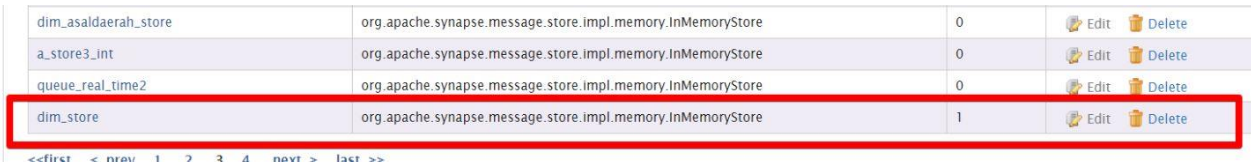
```
<ns1167:create_dim_prodi
    xmlns:ns1167="http://tempuri.org">
    <__numeric_0 xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="unnamed_struct_use_soapval[8]">
        <item>
            <prodi_id xsi:type="xsd:string">1</prodi_id>
            <nama_prodi xsi:type="xsd:string">Teknik Informatika</nama_prodi>
            <akreditasi xsi:type="xsd:string">B</akreditasi>
            <status xsi:type="xsd:string">Aktif</status>
            <konsentrasi xsi:type="xsd:string">Pemrograman</konsentrasi>
        </item>
```

Fig. 4. First pattern request example.



```
<ns1:create_dim_prodiResponse
    xmlns:ns1="http://tempuri.org">
    <return xsi:type="tns:list_dim_prodi">
        <jenis_dt xsi:type="xsd:string">0</jenis_dt>
        <daftar_dim_prodi xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:dim_prodi[8]">
            <item xsi:type="tns:dim_prodi">
                <id_dim_prodi xsi:type="xsd:int">1</id_dim_prodi>
                <nama_prodi xsi:type="xsd:string">Teknik Informatika</nama_prodi>
            </item>
```

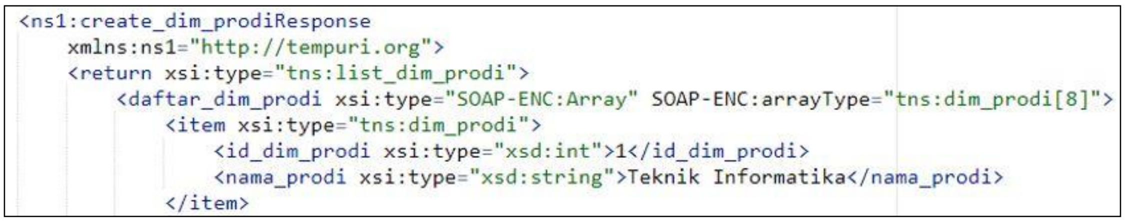
Fig. 5. First pattern service result.



dim_asaldaerah_store	org.apache.synapse.message.store.impl.memory.InMemoryStore	0	Edit Delete
a_store3_int	org.apache.synapse.message.store.impl.memory.InMemoryStore	0	Edit Delete
queue_real_time2	org.apache.synapse.message.store.impl.memory.InMemoryStore	0	Edit Delete
dim_store	org.apache.synapse.message.store.impl.memory.InMemoryStore	1	Edit Delete

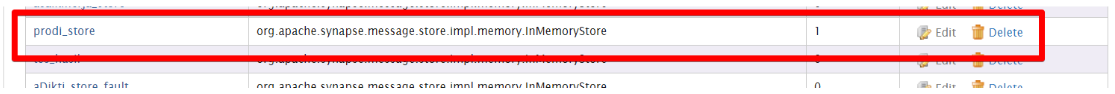
<<first < prev 1 2 3 4 next > last >>

Fig. 6. Result in message store.



```
<ns1:create_dim_prodiResponse
    xmlns:ns1="http://tempuri.org">
    <return xsi:type="tns:list_dim_prodi">
        <daftar_dim_prodi xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:dim_prodi[8]">
            <item xsi:type="tns:dim_prodi">
                <id_dim_prodi xsi:type="xsd:int">1</id_dim_prodi>
                <nama_prodi xsi:type="xsd:string">Teknik Informatika</nama_prodi>
            </item>
```

Fig. 7. Second pattern service result.



prodi_store	org.apache.synapse.message.store.impl.memory.InMemoryStore	1	Edit Delete
aDikti_store_fault	org.apache.synapse.message.store.impl.memory.InMemoryStore	0	Edit Delete

Fig. 8. Second pattern result in message store.

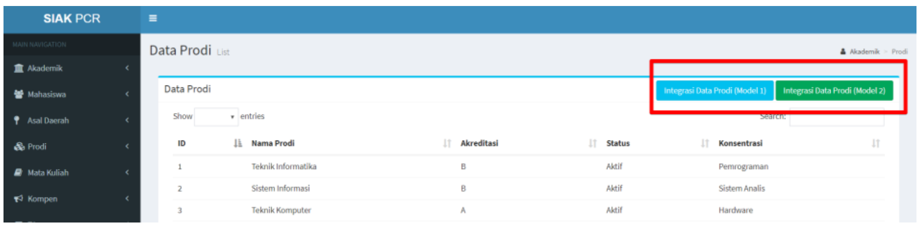
The message is returned and stored in the message store. The message forwarding processor then detects each message in message store and forward it to external service. The service's task is to save the data to data warehouse. After that, the data are inserted to data warehouse as in Figure 12.

B. *Testing*

This research uses SOA testing method based on ISO/IEC 9126 on three aspects: functionality, efficiency, and reliability.

1) *Functionality Testing*

Functionality testing is conducted using black-box testing method with 14 test cases. Based on the result of black-box testing, the functionality aspect can be calculated using formula based on ISO 9126 as shown in (1) and (2) for the first pattern and the second pattern, respectively. Based on the calculation result, both of the integration models fulfill the functionality aspect based on ISO/IEC 9126.

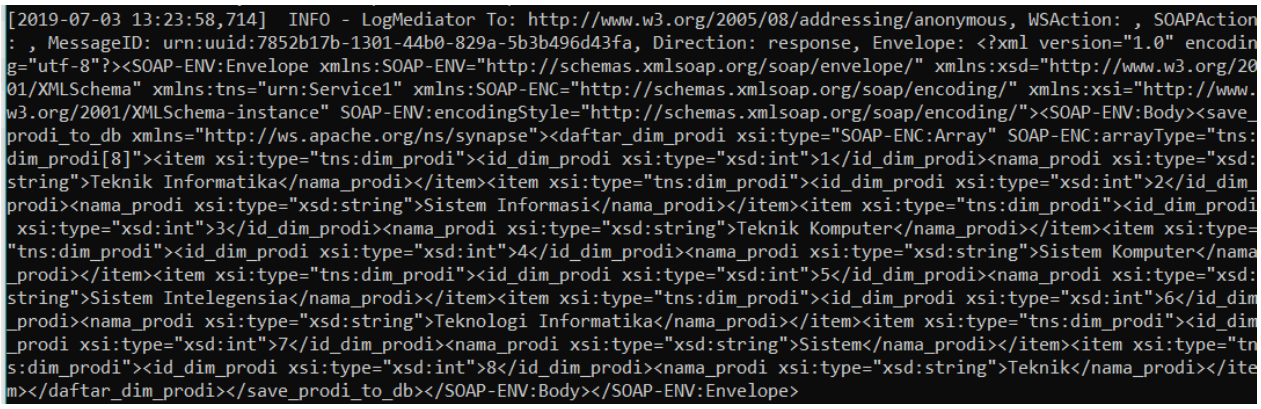


Data Prodi List

Data Prodi

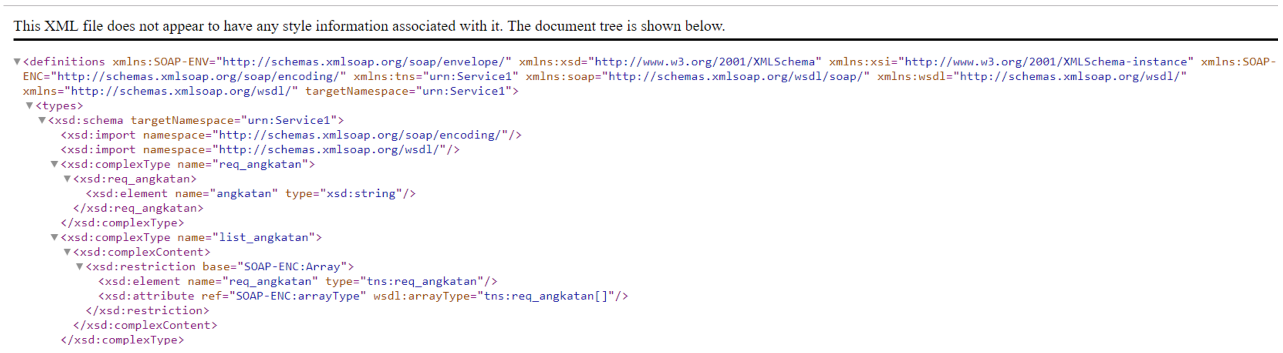
ID	Nama Prodi	Akreditasi	Status	Konsentrasi
1	Teknik Informatika	B	Aktif	Pemrograman
2	Sistem Informasi	B	Aktif	Sistem Analis
3	Teknik Komputer	A	Aktif	Hardware

Fig. 9. Integration process trigger.



```
[2019-07-03 13:23:58,714]  INFO - LogMediator To: http://www.w3.org/2005/08/addressing/anonymous, WSAction: , SOAPAction
: , MessageID: urn:uuid:7852b17b-1301-44b0-829a-5b3b496d43fa, Direction: response, Envelope: <?xml version="1.0" encodin
g="utf-8"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/20
01/XMLSchema" xmlns:tns="urn:Service1" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><save_
prodi_to_db xmlns="http://ws.apache.org/ns/synapse"><daftar_dim_prodi xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:
dim_prodi[8]"><item xsi:type="tns:dim_prodi"><id_dim_prodi xsi:type="xsd:int">1</id_dim_prodi><nama_prodi xsi:type="xsd:
string">Teknik Informatika</nama_prodi></item><item xsi:type="tns:dim_prodi"><id_dim_prodi xsi:type="xsd:int">2</id_dim_
prodi><nama_prodi xsi:type="xsd:string">Sistem Informasi</nama_prodi></item><item xsi:type="tns:dim_prodi"><id_dim_prodi
 xsi:type="xsd:int">3</id_dim_prodi><nama_prodi xsi:type="xsd:string">Teknik Komputer</nama_prodi></item><item xsi:type=
"tns:dim_prodi"><id_dim_prodi xsi:type="xsd:int">4</id_dim_prodi><nama_prodi xsi:type="xsd:string">Sistem Komputer</nama
_prodi></item><item xsi:type="tns:dim_prodi"><id_dim_prodi xsi:type="xsd:int">5</id_dim_prodi><nama_prodi xsi:type="xsd:
string">Sistem Intelegensia</nama_prodi></item><item xsi:type="tns:dim_prodi"><id_dim_prodi xsi:type="xsd:int">6</id_dim
_prodi><nama_prodi xsi:type="xsd:string">Teknologi Informatika</nama_prodi></item><item xsi:type="tns:dim_prodi"><id_dim
_prodi xsi:type="xsd:int">7</id_dim_prodi><nama_prodi xsi:type="xsd:string">Sistem</nama_prodi></item><item xsi:type="tn
s:dim_prodi"><id_dim_prodi xsi:type="xsd:int">8</id_dim_prodi><nama_prodi xsi:type="xsd:string">Teknik</nama_prodi></ite
m></daftar_dim_prodi></save_prodi_to_db></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

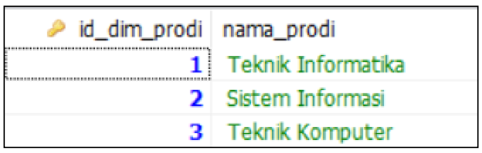
Fig. 10. Middleware incoming message log.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
 ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:Service1" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:Service1">
  ▼<types>
    ▼<xsd:schema targetNamespace="urn:Service1">
       <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
       <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/"/>
     ▼<xsd:complexType name="req_angkatan">
       ▼<xsd:req_angkatan>
           <xsd:element name="angkatan" type="xsd:string"/>
         </xsd:req_angkatan>
       </xsd:complexType>
     ▼<xsd:complexType name="list_angkatan">
       ▼<xsd:complexContent>
         ▼<xsd:restriction base="SOAP-ENC:Array">
             <xsd:element name="req_angkatan" type="tns:req_angkatan"/>
             <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:req_angkatan[]"/>
           </xsd:restriction>
         </xsd:complexContent>
       </xsd:complexType>
```

Fig. 11. Service's WSDL.



id_dim_prodi	nama_prodi
1	Teknik Informatika
2	Sistem Informasi
3	Teknik Komputer

Fig. 12. Data warehouse result.

A_1 = 0 (total failed functionality test cases for first pattern)
B_1 = 14 (total success functionality test cases for first pattern)

$$X_1 = 1 - \frac{0}{14} = 1 \tag{1}$$

A_2 = 0 (total failed functionality test cases for second pattern)
B_2 = 14 (total success functionality test cases for second pattern)

$$X_2 = 1 - \frac{0}{14} = 1 \tag{2}$$

2) Performance Testing

The aspects tested for performance aspects are the messaging patterns' runtime performance to different load of clients' requests. Each of the patterns is compared, where the fastest pattern can be said more efficient. The result is obtained using Apache JMeter tools and shown as response time graph.

Figure 13 shows a graph of benchmarking result for sending fact table ips, where blue line shows 2nd pattern result and red line shows 1st pattern result.

Based on the efficiency testing conducted on Apache JMeter, it is shown that the 2nd pattern is faster in reformatting data from client request and sending data to external service.

3) Reliability Testing

The scenario is to test the reliability of the messaging procedure in sending the message from client to data warehouse when the data warehouse server is turned down. The messaging pattern is supposed to be able to handle server's unavailability. In this situation, the messaging pattern holds the responsibility to ensure the data insertion successful. The messaging pattern can be said reliable if the data is inserted successfully and the data is complete.

Based on the reliability testing that has been conducted on 10 test cases, the reliability aspect can be obtained using formula based on ISO 9126 as shown in (3) and (4) for the first and the second patterns respectively.

$A1_1$ = 0 (the total number of failures were detected during a defined trial period)
$A2_1$ = 10 (the number of executed test cases)

$$X_1 = \frac{0}{10} = 0 \tag{3}$$

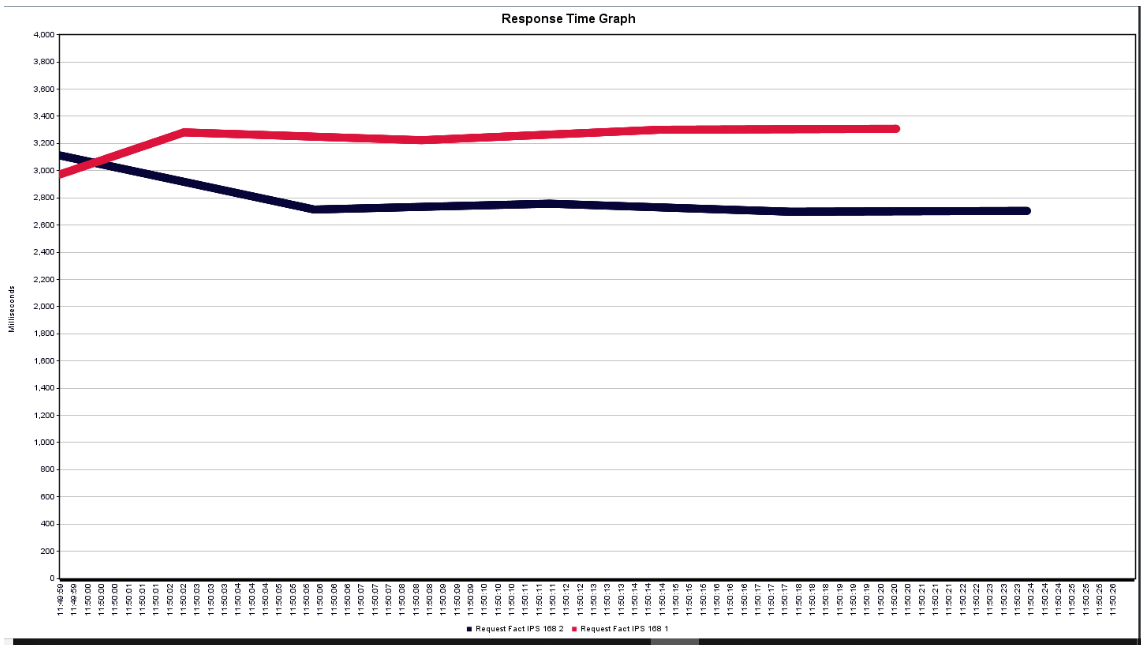


Fig. 13. Efficiency testing result.

$A1_2$ = 0 (the total number of failures were detected during a defined trial period)
$A2_2$ = 10 (the number of executed test cases)

$$X_2 = \frac{0}{10} = 0 \tag{4}$$

As both of the reliability aspect scores are equal 0, both patterns can be said as reliable.

C. Analysis

Based on testing that has been conducted referring to ISO 9126, the result shows that both integration model fulfill the functionality and reliability aspect. The integration model can be said successfully meet functionality requirement when the functionality score close to one. Both integration models have functionality metric score equal to one meaning that both integration models meet the requirement. For reliability aspect, the integration models meet the requirement from reliability metrics score, which is closer to 0. Both integration models score 0 for reliability aspect.

For the performance aspect, as shown from the benchmarking result, the 2nd pattern has better performance on most of the test cases. The 1st pattern only excels on 2 test cases: dim “asaldaerah” integration process and fact “kelulusan” integration process. Seeing that 9 out of 11 test cases show that the 2nd pattern has faster average response time, it can be concluded that the 2nd pattern is more efficient.

From the testing result, it can be concluded that the 2nd integration model is a better model to integrate system in near-real time. This can be achieved by having different message channel, message store, and message processor for each dimension and fact table. When there are multiple incoming requests from client, in the 1st integration model, the request is processed only by 2 message channels and message stores. While the middleware is still processing the message, another message is queued within the same message store, elongating the message queue within message store. On the other hand, the 2nd pattern distributes incoming requests to different message stores. The queue for each message processor is shortened than in the 1st pattern. This condition makes the response time become slower gradually for the 1st pattern. This condition, however, mostly can be observed with multiple loads of requests.

Another cause of better performance with the 2nd pattern is because the 2nd pattern does not require data identifier within the message, while the 1st pattern does. The incoming message on the 1st pattern is only handled by 1 message channel, thus require data identifier to differentiate each incoming message type. Putting and reading data identifier require more time, making the 1st pattern has slower response time.

V. Conclusion and Suggestion

A. Conclusion

Based on the implementation and testing that has been conducted, the comparison of the pattern result can be concluded as follows. Using the Enterprise Application Integration and utilizing message-oriented middleware to implement data integration can integrate data from and to different systems in near-real time. Both data integration models are able to provide sufficient data in the fitting format to data warehouse for the business intelligence system. Both data integration models can ensure the reliability of data in data warehouse by using the data-loss prevention within message-oriented middleware. Based on benchmarking results, it’s better to distinct the message channel and message store of data within middleware, especially when it typically has many requests. This way, it can enhance the performance of data integration model.

B. Suggestion

For future development, there are several suggestions as follows. (1) For better implementation, the middleware should able to send both HTTP 202 code first to client and HTTP 200 code after the actual request has finished as information to the client-side. This might need further research using I/O socket as HTTP request can only accept one request and one response. (2) The current model hasn’t been equipped with fault sequence to prevent problems within middleware. It’s best to provide middleware with fault sequence as error handling method within middle-ware.

References

[1] A. P. Sani, “Pengembangan Sistem Business Intelligence untuk Analisis Kinerja Akademik Mahasiswa di Lingkungan Jurusan Teknologi Informasi Politeknik Caltex Riau,” Politeknik Caltex Riau, 2018.

[2] E. Marshila, “Sistem Informasi Apotek Menggunakan Teknologi Enterprise Service Bus (ESB) (Studi Kasus : Apotek Rumbai Sehat),” *J. Aksara*

Komput. Terap. Politek. Caltex Riau, vol. 2, no. 1, 2013.
[3] I. M. D. J. Sulastra, M. Sudarma, and I. N. S. Kumara, "Pemodelan Integrasi Nearly Real Time Data Warehouse dengan Service Oriented Architecture Untuk Menunjang Sistem Informasi Retail," *Maj. Ilm. Teknol. Elektro*, vol. 14, no. 2, p. 11, 2015.
[4] A. A. G. Y. Paramartha, N. K. Kertiasih, and G. R. Dantes, "Integrasi Data Penelitian , Pengabdian kepada Masyarakat , dan Kinerja Dosen di Universitas Pendidikan Ganesha," *Semin. Nas. Vokasi dan Teknol.*, pp. 314–320, 2016.
[5] A. A. Ali and W. M. Mohamed, "Using a Business Activity Monitoring and SOA for a Real-Time ETL," *Int. J. Comput. Appl.*, vol. 180, no. 10, pp. 975–8887, 2018.
[6] A. Halevy and J. Ordille, "Data Integration : The Teenage Years," *Artif. Intell.*, vol. 41, no. 1, pp. 9–16, 2006.
[7] M. A. P. M. Van Den Bosch, M. E. Van Steenbergen, M. Lamaitre, and R. Bos, "A Selection-Method for Enterprise Application," *Perspect. Bus. Informatics Res. 9th Int. Conf. BIR 2010, Rostock Ger. 2010. Proc.*, vol. 9th Intern, pp. 176–187, 2010.
[8] A. Wibowo, "Problems and Available Solutions on the Stage of Extract, Transform, and Loading in Near Real-Time Data Warehousing (A Literature Study)," *2015 Int. Semin. Intell. Technol. Its Appl. ISITIA 2015 - Proceeding*, pp. 345–349, 2015.
[9] I. S. Gracia, A. Wibowo, and I. Muslim, "Rancang Bangun Aplikasi E-commerce dengan Mengotomatisasi Purchase Order (PO) Menggunakan Service Oriented Architecture dan Metode Fast Slow Non (FSN) Analysis," *J. Aksara Komput. Terap.*, vol. 5, no. 1, 2016.
[10] E. Curry, "Message-Oriented Middleware," in *Middleware for Communication*, Q. H. Mahmoud, Ed. Chicester: Wiley, 2004, pp. 1–28.
[11] B. Woolf and G. Hohpe, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston: Addison-Wesley, 2004.