Ethos (Jurnal Penelitian dan Pengabdian Masyarakat): 63-68

PROTOTIPE *WEB SERVICE* SISTEM PENGIRIMAN BARANG DENGAN PENERAPAN *ENTERPRISE SERVICE BUS*

THE WEB SERVICE PROTOTYPE ON DELIVERY SYSTEM IN THE IMPLEMENTATION OF ENTERPRISE SERVICE BUS

Ghifari Munawar

Jurusan Teknik Komputer dan Informatika, Politeknik Negeri Bandung, Jl. Gegerkalong Hilir, DsCiwaruga, Bandung 40012

e-mail: ghifari.munawar@polban.ac.id

Abstract. *The main component of the logistics system is a delivery goods system. It has an enormous role in managing the entire historical shipment data from the start point (origin) to the end of delivery (destination). This research aims to implement the Enterprise Service Bus (ESB) on delivery systems as a middleware in the integration data process. ESB technology used in this research is NServiceBus. The stages of research using a prototype model to develop a web service that suits with theirs needs. Testing is done by tested two aspects of the exchange messages; the performance aspect and the aspect of independence. The test results show that the performance of the web service with the ESB application is better than the non-ESB user and Web services developed to have a good level of independence (loosely coupling).*

Keywords: Service Oriented Architecture (SOA), Enterprise Service Bus (ESB), NServiceBus, Delivery Good System

Abstrak. *Komponen utama dari sistem logistik adalah sistem pengiriman barang. Perannya sangat besar dalam mengelola seluruh historis data pengiriman mulai dari titik awal (origin) hingga titik akhir pengiriman (destination). Penelitian ini bertujuan untuk menerapkan enterprise service bus (ESB) pada sistem pengiriman barang sebagai middleware dalam proses integrasi data. Teknologi ESB yang digunakan pada penelitian ini adalah NServiceBus. Tahapan penelitian menggunakan model prototype untuk mengembangkan web service yang sesuai dengan kebutuhan. Pengujian dilakukan dengan menguji dua aspek pertukaran pesan, yakni aspek performa dan aspek independensi. Hasil pengujian menunjukan bahwa performa web service dengan penerapan ESB lebih baik dibandingkan Non ESB serta web service yang dikembangkan memiliki tingkat independensi yang baik (loosely coupling).*

Kata kunci: Service Oriented Architecture (SOA), Enterprise Service Bus (ESB), NServiceBus, Sistem Pengiriman Barang

1. Pendahuluan

Industri penyediaan jasa logistik di Indonesia telah berkembang pesat dan diperkirakan masih akan terus bertumbuh. Hal ini dapat didasari pula oleh pertumbuhan ekonomi kreatif yang mulai meningkat, para pebisnis mulai memperluas target pasarnya dengan memasarkan produk secara online, baik dengan mengelola website *e-commerce* sendiri, maupun melalui website *market place*. Kebutuhan akan jasa logistik yang terus meningkat tentunya harus sejalan dengan pengelolaan sistem yang baik, karena salah satu poin penting yang menjadi pertimbangan perusahaan dalam memilih jasa logistik adalah kehandalan teknologi informasinya (Sudrajat, 2010). Besarnya data yang dimiliki sistem logistik di perusahaan dapat menjadi hambatan dalam proses

operasionalnya apabila tidak dikelola dengan baik, terutama untuk sistem logistik yang menerapkan akses pertukaran data melalui *web service*.

Komponen utama *supply chain management* (SCM) dari sistem logistik adalah sistem pengiriman barang. Perannya sangat besar dalam mengelola seluruh historis data pengiriman, mulai dari awal pengiriman di lokasi asal (*origin*) hingga akhir pengiriman ke lokasi tujuan (*destination*). Terdapat beberapa komponen lain dalam sistem logistik, diantaranya adalah sistem pengelolaan armada, sistem pengelolaan kurir, sistem manajemen rute pengiriman, sistem monitoring, sistem CRM (*Customer Relationship Management*), dll. Permasalahan yang mungkin timbul dari banyaknya komponen dalam sistem logistik adalah kesulitan dalam proses integrasi data, karena kompleksitas yang disebabkan oleh implementasi arsitektur sistem yang berbeda dan *platform* bahasa pemrograman yang berbeda pula (heterogen). Selain itu mekanisme pertukaran data yang dilakukan secara *point to point* antara *service consumer* ke *service provider* (*tightly coupling*) akan menyebabkan beban *provider* menjadi besar.

Service Oriented Architecture (SOA) merupakan sekumpulan layanan teknologi, arsitektur, dan prinsip desain terkait yang dapat diimplementasikan pada lingkungan komputasi yang berbeda. Teknologi SOA dapat menghubungkan satu komponen dengan komponen lain dengan tetap menjaga independensinya (*loosely coupling*), serta memungkinkan untuk cepat beradaptasi terhadap perubahan proses bisnis (Beydoun, 2013). Teknologi SOA memiliki beberapa komponen, salah satunya adalah *Enterprise Service Bus* (ESB). ESB dapat dikatakan sebagai *middleware* dalam proses integrasi data yang menghubungkan antara *service provider* dengan *service consumer*, sehingga dapat menghilangkan ketergantungan akses komunikasi secara langsung / *point-to-point* nya. ESB merupakan infrastruktur untuk koneksi layanan SOA dan mekanisme pertukaran pesannya. Fungsionalitas utama ESB adalah melakukan rute, transformasi protokol serta transformasi pesan atau data (Juric, 2010). Dengan demikian, ESB dapat meningkatkan *reusability* dari *service* yang dikembangkan dan dapat menghasilkan sistem yang mudah beradaptasi terhadap perubahan kebutuhan layanan.

ESB telah banyak diimplementasikan dalam membangun mekanisme pertukaran data pada sistem skala *enterprise*, salah satunya adalah penerapan ESB pada sistem pengelolaan bandara. Berdasarkan hasil penelitian (Suha, 2013), ESB dapat meningkatkan aksesibilitas dan *sharing* data antar departemen di bandara, mengubah layanan tanpa harus memodifikasi kode program, serta memaksimalkan keandalan sistem. Penelitian ini akan mengembangkan prototipe sistem pengiriman barang berbasis SOA dengan penerapan ESB. Tujuan dilakukannya penelitian ini adalah sebagai berikut : (1) merancang arsitektur *web service* pada sistem pengiriman barang dengan penerapan ESB, (2) membangun prototipe berdasarkan hasil rancangan, (3) menguji performa dan independensi *service* yang telah dibangun. Dengan penerapan ESB pada sistem pengiriman barang, integrasi data antar modulnya akan lebih efektif, dan ketika terjadi perubahan proses bisnis, independensinya akan tetap terjaga (*loosely coupling*). Sehingga dapat meningkatkan performa sistem dan secara tidak langsung juga dapat meningkatkan efektifitas dan efesiensi operasional pengiriman barang pada perusahaan logistik.

2. Metodologi Penelitian

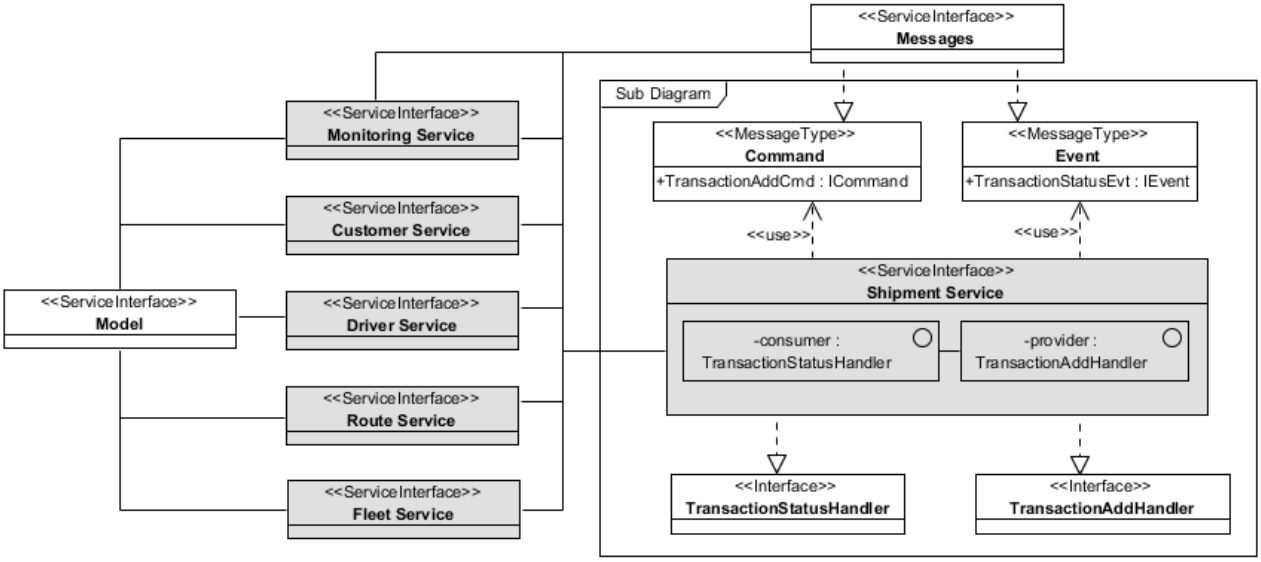
Penelitian ini dilakukan pada sistem pengiriman barang yang telah dibangun, untuk kemudian di *re-engineering* dengan penerapan ESB. Sistem pengiriman barang yang dijadikan objek penelitian ini dikembangkan pada platform *ASP.NET MVC* dengan database *SQL Server* dan memiliki enam modul utama, yaitu : (1) modul pengelolaan

armada, (2) modul pengelolaan kurir, (3) modul pengelolaan pelanggan, (4) modul pengelolaan transaksi pengiriman, (5) modul pengelolaan rute pengiriman, dan (6) modul monitoring pengiriman. Masing-masing modul kemudian dianalisa perilaku sistemnya untuk kemudian dirancang prototipe *web service*nya. Tahapan penelitan dilakukan dengan metodologi *prototype*, dimana pada tahap analisa, perancangan, implementasi, dan pengujian dilakukan secara *iterative* sampai menemukan prototipe *web service* yang sesuai dengan kebutuhan sistem.

3. Hasil dan Pembahasan

Prototipe *Web Service* dengan Penerapan ESB

Teknologi ESB yang digunakan pada penelitian ini adalah *NServiceBus*, dimana media pengelolaan untuk pertukaran pesan dan antriannya menggunakan MSMQ (*Message Queue*). *NServiceBus* membagi pesan kedalam dua tipe, yaitu (1) *command*, dan (2) *event*. Prototipe *web service* dirancang sesuai dengan modul yang ada pada sistem, sedangkan pengaturan pesan disesuaikan dengan perilaku layanan pada sistem yang berjalan. Prosedur layanan yang mengirimkan pesan hanya ke satu modul penerima dikategorikan sebagai *command*, dan prosedur layanan yang mengirimkan pesan ke lebih dari satu modul penerima dikategorikan sebagai *event*. Berikut ini adalah gambar arsitektur *web service* yang telah dikembangkan dengan penerapan ESB :



Gambar 1. Arsitektur *Web Service Pada* Sistem Pengiriman Barang Dengan *NserviceBus*

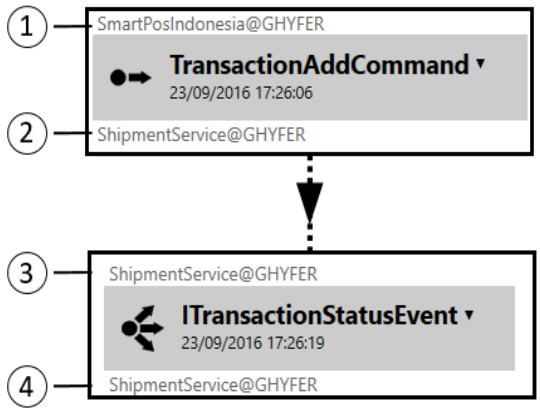
Seperti terlihat pada Gambar 1, terdapat enam *service* utama yang dikembangkan, yaitu : (1) *Fleet Service*, (2) *Driver Service*, (3) *Customer Service*, (4) *Shipment Service*, (5) *Route Service*, dan (6) *Monitoring Service*, serta terdapat beberapa *service* lain yang dibutuhkan, yaitu *Messages* untuk menampung pesan, dan `Model` untuk layanan ke database. Dapat dilihat perilaku prosedur layanan yang berjalan pada subdiagram di Gambar 1, ketika *service consumer* mengirimkan pesan (*command*) `TransactionAddCmd` untuk menambahkan transaksi pengiriman barang, maka pesan tersebut akan diteruskan ke *service provider* (*Shipment Servicer*), dimana terdapat kelas `TransactionAddHandler` untuk memproses pesan tersebut. Setelah diproses pesan tersebut kemudian di-*publish* untuk menjadi *event* (`TransactionStatusEvt`) yang kemudian diproses kembali oleh *subscriber*, yaitu kelas `TransactionStatusHandler`. Masing-masing *web service* dapat mengirimkan pesan pada *web service* lain yang

dibutuhkan, pertukaran pesan ini berjalan secara *asynchronous*. Hal ini tidak akan membebani sistem ketika akses *consumer* terhadap *provider* dilakukan secara kontinyu, karena setiap pesan yang dikirim akan dikelola ke dalam antiran (MSMQ), sehingga performa *service* akan tetap terjaga.

Dalam mengimplementasikan *Shipment Service*, dibuat kelas `TransactionAddCommand` sebagai *command* yang digunakan untuk mengirimkan data transaksi pengiriman barang dan *interface* `ITransactionStatusEvent` sebagai *event* yang akan di-*publish* kepada *service* lain ketika terjadi perubahan status transaksi pengiriman. Dalam mengirimkan *command* kepada *webservice*, method *NServiceBus* yang digunakan adalah `Bus.Send()`, sedangkan `Bus.Publish<event>` digunakan untuk mem-*publish event*. Proses yang terjadi ketika pesan `TransactionAddCommand` dikirimkan adalah (1) menambahkan transaki pengiriman ke database, (2) mengubah status transaksi, (3) mengirimkan notifikasi. Dapat dilihat pada tabel 1 di bawah ini terkait implementasi *Shipment Service* dengan *NServiceBus* :

Table 1. Implementasi *Shipment Service* dengan *NServiceBus*

Flow Diagram		Source Code

	1	`...`
		`var cmd = new Messages.Commands.TransactionAddCommand`
		`{`
		`  ShipmentTransaction = shipmentTransaction`
		`};`
		`ServiceBus.Bus.Send("ShipmentService", cmd);`
		`...`

	2	`public class TransactionAddCommandHandler:IHandleMessages<TransactionAddCommand>`
		`{`
		`  ...`
		`Bus.Publish<ITransactionStatusEvent>(evt =>`
		`  {`
		`  evt.TransactionId = msg.TransactionId;`
		`  evt.Status = "Manifested";`
		`  }`
		`  ...`
		`}`

	3	`public class TransactionStatusEventHandler : IHandleMessages<ITransactionStatusEvent>`
		`{`
		`  public void Handle(ITransactionStatusEvent message)`
		`  {`
		`  ...`
		`  Log.InfoFormat("Status Transaction Id " + message.TransactionId + " changed to " + message.Status);`
		`  }`
		`}`

	4	`public class NotificationEventHandler : IHandleMessages<ITransactionStatusEvent>`
		`{`
		`  ...`
		`  public void Handle(ITransactionStatusEvent message)`
		`  {`
		`  ...`
		`  Log.InfoFormat("Email Sent to " + shipmentTransaction.ConsigneeName + ", Transaction status is " + message.Status);`
		`  }`
		`}`

Ketika *command* dikirimkan melalui *controller* ke *Shipment Service*, kelas TransactionAddCommandHandler kemudian memproses *command* tersebut dan mem-*publish*

ke event ITransactionStatusEvent. Kelas-kelas yang menjadi *subscriber*nya adalah TransactionStatusEventHandler dan NotificationEventHandler, dimana masing-masing kelas akan memproses *event* yang telah di-*publish* tersebut. Berikut ini adalah log aktifitas yang tercatat ketika pesan diproses :

```
2016-09-23 20:52:13.920 INFO  ShipmentService.TransactionAddCommandHandler Shipment with Transaction Id : TR23091601 Has been Added

2016-09-23 20:52:23.874 INFO  Messages.Events.ITransactionStatusEvent Email Sent to Anjas, Transaction status is Manifested

2016-09-23 20:52:24.376 INFO  Messages.Events.ITransactionStatusEvent Status Transaction Id TR23091601 changed to Manifested
```

Gambar 2. Log *NServiceBus* ketika *Shipment Service* memproses pesan

Pengujian Web Service

Prototipe *web service* yang telah dikembangkan kemudian diuji dalam 2 aspek, yaitu (1) aspek performa, (2) aspek independen. Aspek performa diukur dengan menghitung waktu respon sistem saat memproses sebuah pesan antara *service* yang menerapkan ESB dan yang tidak (Non ESB), sedangkan aspek independen diukur dengan mengubah perilaku layanan pada *provider* dalam memproses permintaan layanan tanpa mengubah kode program dari *consumer*. Spesifikasi komputer server yang digunakan dalam uji coba adalah sebagai berikut : (1) *Processor* : *Quad-Core 1 GHz*, (2) *RAM* : 4 *GB*, (3) *Harddisk* : 128 *GB SSD*, (4) Sistem Operasi : *Windows 10*. Berikut ini adalah hasil pengujian performa *web service* saat memproses pesan :

Table 2. Pengujian Performa *Web Service*

No	Web Service	Method	Waktu Respon (ms)	
			ESB	Non ESB
1	*CustomerService*	POST	747.13 ms	1723.35 ms
2	*ShipmentService*	POST	381.56 ms	2046.57 ms
3	*DriverService*	POST	944.23 ms	1868.73 ms
4	*FleetService*	POST	384.66 ms	557.04 ms
5	*MonitoringService*	GET	261.12 ms	378.1 ms

Tabel 2 menunjukan bahwa waktu respon sistem pada *web service* yang menerapkan ESB lebih cepat dibandingkan dengan Non ESB. Dapat dilihat pada baris ke-1 untuk pengujian *CustomerService*, ketika memproses suatu pesan dalam menambahkan data *customer*, waktu respon yang dibutuhkan adalah 747.13 ms dengan penerapan ESB, sedangkan Non ESB membutuhkan waktu respon sebesar 1723.35 ms. Begitupun dengan pengujian pada *web service* lainnya, hasil pengujian menunjukan bahwa performa *web service* dengan penerapan ESB lebih baik dibandingkan Non ESB.

Pada pengukuran aspek independen, peneliti melakukan beberapa percobaan untuk mengubah perilaku layanan *web service* dengan mengubah kode program dan alur lojik dari *service provider*. Ketika diuji untuk mengirim suatu pesan tanpa mengubah kode program dari *consumer*, *provider* tetap memprosesnya sesuai dengan layanan yang diubah. Hal ini disebabkan dalam penerapan ESB, layanan yang disediakan oleh *provider* tidak secara langsung diakses oleh *consumer* melainkan melalui pertukaran pesan. Pesan inilah yang menjadi media komunikasi antara *provider* dengan *consumer*. Sehingga hal ini membuktikan bahwa arsitektur ESB yang dikembangkan memiliki tingkat independensi yang baik (*loosely coupling*).

4. Kesimpulan

Berdasarkan penelitian yang telah dilakukan dapat diambil kesimpulan sebagai berikut : (1) Arsitektur *web service* dengan penerapan ESB telah dirancang sesuai dengan kebutuhan layanan; (2) Prototipe *web service* yang telah dibangun, diantaranya : *Fleet Service*, *Driver Service*, *Customer Service*, *Shipment Service*, *Route Service*, *Monitoring Service*, *Messages*, dan *Model*; (3) Hasil pengujian menunjukan bahwa performa *web service* dengan penerapan ESB lebih baik dibandingkan Non ESB serta *web service* yang dikembangkan memiliki tingkat independensi yang baik (*loosely coupling*).

Saran untuk pengembangan penelitian selanjutnya adalah dengan mengkaji beberapa pola pertukaran pesan pada ESB. ESB dengan teknologi *NServiceBus* memiliki pola integrasi yang beragam, diantara pola *publish-subscribe*, *request-reply*, *databus*, *timeout*, *message mutations*, *message encryption*, *scale out*, dan saga. Penelitian selanjutnya dapat meneliti berbagai pola integrasi tersebut dan mengujinya untuk dikaji hasilnya, pola mana yang paling sesuai dengan kebutuhan layanan apa, sehingga memungkinkan untuk mendapatkan mekanisme integrasi yang lebih baik.

Ucapan Terima Kasih

-

Daftar pustaka

Sudrajat, Darjat. (2010). Segmentasi Pasar Penyedia Jasa Logistik (Third Party Logistics) di Indonesia. Fakultas Ekonomi dan Bisnis. Universitas Bina Nusantara. Jakarta Barat.

Beydoun, Ghassan, dkk. (2013). Service Oriented Architectures (SOA) Adoption Challenges. School of Information System and Technology. University of Wollongong. Australia. International Journal of Intelligent Information Technologies. 9(2), 1-6, April-June.

Endrei, Mark, dkk. (2004). Patterns : Service Oriented Architecture and Web Services. RedBooks IBM.

Afaneh, Suha (2013). Airport Enterprise Service Bus with Three Levels Self-Healing Architecture (AESB-3LSH). Departement of Computer Science, Faculty of Information Technology, Isra University. Jordan.

Juric, M.B., Chandrasekaran, S., Frece, A., Hertis, M., dan Srdic, G. (2010). WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, Packt Publishing Ltd. 32 Lincoln Road Olton Birmingham, B27 6PA, UK.

Hullu, Clermond de. (2012). Evaluating .NET-Based Enterprise Service Bus Solution. Departement of Computer Science. University of Twente.

Carreraz, M. Antonia Martinez, dkk. (2012). Enterprise Service Bus for Building Integrated Enterprises. Universidad de Murcia, Spain.

Boike, David. (2015). *Learning NServiceBus Second Edition – Building reliable and scalable distributed software systems using the industry leading .NET Enterprise Service Bus*. Packt Publishing. Birmingham, UK.

Brall, Prashant. (2016). *NServiceBus Integration Pattern – Part 1*. [Online]. Tersedia : https://prashantbrall.wordpress.com/tag/nservicebus/. [15 Maret 2016]

Utomo, Wiranto Herry. (2012). Penerapan Enterprise Service Bus (ESB) Sebagai Middleware Integrasi Berbasis SOA. Makalah Ilmiah Seminar Nasional Teknologi Informasi dan Komunikasi (SENTIKA). Yogyakarta.