KINERJA ARSITEKTUR INTEROPERABILITAS E-GOVERMENT MULTI *PLATFORM*

Aris Puji Widodo
Jurusan Ilmu Komputer/Informatika FSM Unviversitas Diponegoro
Jl.Prof. H.Soedarto,SH, Tembalang, Semarang

Abstract. The implementation of e-Government (e-Gov)The implementation of e-Government (e-Gov) to produce a form of relationship Government to Government (G2G). The main factor G2G about interoperability between national and local governments that have a development platform heterogeneity of e-Gov. Interoperability happens is done by creating a new architecture that has the function so that the E-Gov existing ones can communicate with each other. E-Gov Indonesia still developed separately between the central (national) and regional (province, district) or between agencies, so interoperability is happening yet fully able to walk properly.
This Paper is focused on making application interoperability architecture of the E-Gov Indonesia as case studies in order to model the heterogeneous platform interoperability scheme. Architecture developed using the approach to Services Oriented Architecture (SOA) in the form of services layer to perform interoperability between heterogeneous platforms. Performance measurement is done by measuring the length of time the execution of the services contained on this architecture by using varying amounts of data.

Keywords: e-Gov, Interoperability, Multiple Platforms, SOA

1. PENDAHULUAN

E-Gov merupakan bentuk pemanfaatan Information Technology (IT) yang digunakan untuk meningkatkan hubungan antara pemerintah pusat dan daerah, antar daerah serta antara pemerintah dengan pihak-pihak lain, terutama dengan warga negara [1]. Penerapan e-Gov dapat menghasilkan sebuah bentuk hubungan seperti: Government to Citizen (G2C), Government to Business (G2B), dan Government to Government (G2G) [2].

Untuk kasus di Indonesia, Implementasi penerapan e-Gov pada level persiapan (pembuatan web site) sudah tergolong sangat baik, artinya hampir setiap pemerintah daerah, pusat, dan lembaga pemerintah memiliki web site dan proses update informasinya juga berlangsung secara terus menerus [3]. Pembuatan web site pada pemerintah daerah , pusat, atau lembaga pemerintah adalah sebagai bentuk untuk menciptakan hubungan G2C dalam hal penyebaran informasi yang lebih terbuka kepada warga negara. Untuk level transaksi dan transformasi masih terfokus pada pengembangan secara terpisah-pisah dengan menggunakan berbagai macam variasi platform pengembangan [3][4]. Untuk proses interoperabilitas yang terjadi antara pemerintah daerah, pusat, dan lembaga pemerintah masih dilakukan dengan cara mempertukarkan media penyimpanan data secara fisik, seperti disket, compact disk dan media penyimpanan yang lain [5]. Kondisi ini dapat diartikan bahwa bentuk hubungan G2G pada e-Gov di Indonesia belum berjalan secara baik, dan belum memenuhi semangat kebijakan pada INPRES No. 3 Tahun 2003 dan blue print sistem aplikasi E-Gov [6][7].

Bentuk hubungan G2G salah satu faktor utamanya adalah mengenai interoperabilitas antara pemerintah daerah dengan pusat, sehingga dapat terjadi pertukaran, pengumpulan, dan integrasi data antara pemerintah daerah dengan pusat [8]. Isu dan tantangan di dalam melakukan interoperabilitas adalah heterogenitas aplikasi e-Gov yang sekarang ini ada. Heterogenitas aplikasi e-Gov yang ada dapat meliputi teknologi, arsitektur, dan platform yang digunakan pada masing-masing

pemerintah daerah berdasarkan kemampuan dan pengetahuan yang dimiliki. Untuk dapat menghasilkan interoperabilitas dari heterogenitas sistem yang ada, pendekatan yang digunakan adalah membangun sebuah arsitektur interoperabilitas multi platform yang dapat mendukung keberadaan arsitektur-arsitektur sebelumnya [9]. Pendekatan arsitektur interoperabilitas multi platform ini bertujuan untuk menghubungkan antar aplikasi e-Gov yang terpisah dan terisolasi untuk dapat berkomunikasi antara satu dengan yang lainnya [10].

Arsitektur interoperabilitas multi platform dikembangkan dengan pendekatan berbasis protokol dan skema *Extensible Markup Language* (XML) yang didistribusikan pada jaringan publik [11][12]. Arsitektur *framework* berbasis protokol dan skema XML ini, proses interoperabilitas multi *platform* yang terjadi hanya dilakukan dengan cara mempertukarkan skema XML yang bersifat open standard dan tidak menyediakan *services*. Untuk menambahkan *services* pada arsitektur *framework* interoperabilitas yang berbasis XML, maka digunakan pendekatan berbasis *Service Oriented Architecture* (SOA) dengan menggunakan teknologi web *services* [10][13][14][15][16][17][18]. *Services* yang disediakan pada arsitektur dengan pendekatan SOA tidak bersifat *real time* terhadap proses bisnis karena *services* yang disediakan hanya berdasarkan pada sebuah *information request and response*, serta bersifat *synchronous messaging* [19].

Untuk menyediakan *services* yang bersifat *real time* terhadap proses bisnis, maka makalah ini difokuskan pada pembuatan arsitektur interoperabilitas multi *platform* dengan menggunakan pendekatan berbasis SOA yang dikombinasikan dengan *Business Process Execution Language* (BPEL). Pendekatan SOA yang digunakan adalah berdasarkan pada konsep pembuatan arsitektur yang dapat melakukan komunikasi dari berbagai macam platform heterogen aplikasi e-Gov yang sekarang ini ada dengan direpresentasikan ke dalam sebuah *services* yang bersifat open standart. BPEL yang digunakan adalah berdasarkan pada konsep untuk penyediaan *services* yang bersifat dinamik. Kombinasi pendekatan SOA dan BPEL bertujuan untuk menyediakan *services* yang dapat bersifat *real time* terhadap proses bisnis, karena *services* yang tersedia dikombinasikan dengan *event* sebagai *trigger* untuk pengiriman messages yang bersifat *asynchronous messaging*.

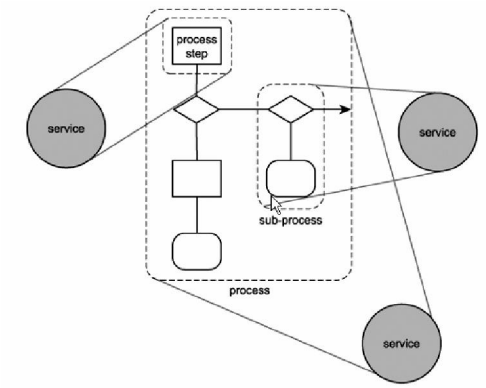
2. PEMBAHASAN

2.1 *Service Oriented Architecture* (SOA)

SOA adalah sebuah bentuk arsitektur teknologi yang mengacu pada prinsip berorientasi pada *service* [20]. Gambar 2.1 mendeskripsikan service pada lingkup SOA merupakan sekumpulan fungsi, prosedur, atau proses yang memberikan *response*, jika diminta oleh sebuah *client* atau dapat juga dipandang sebagai enkapsulasi lojik dari satu atau sekumpulan aktivitas tertentu sebagai implementasi proses bisnis. Konsep yang berorientasi *service* melakukan pendekatan dengan membagi masalah besar menjadi sekumpulan *service* kecil-kecil, dan solusi dari permasalahan tersebut harus dapat diselesaikan dengan memungkinkan seluruh *service* berpartisipasi dalam sebuah orkestrasi. SOA tidak terkait dengan suatu teknologi tertentu, tetapi lebih ke arah pendekatan secara modular.

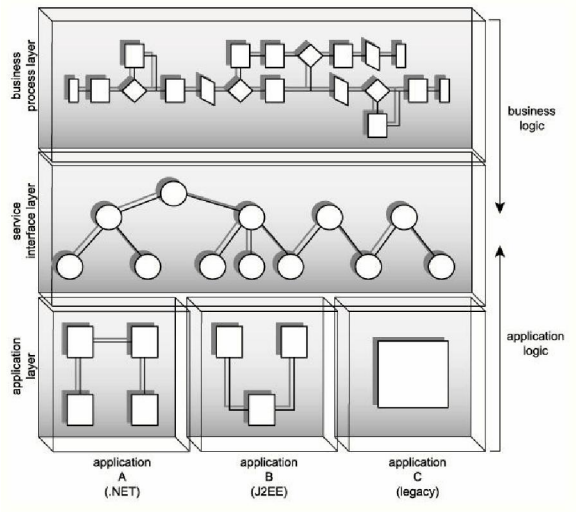
Untuk mengimplementasikan aplikasi dengan pendekatan SOA yang berorientasi *services*, maka diimplementasikan pada sebuah layer yang terletak di antara *business process layer* dan *application layer* yang merupakan bagian dari *enterprise logic*, yang disebut *service interface layer*, seperti pada Gambar 2.2 *Service interface layer* memiliki fungsi untuk melakukan enkapsulasi lojik yang terdapat pada *application logic*, sekaligus proses bisnis yang terdapat pada business logic, sehingga, aplikasi dapat lebih modular dan dapat menggunakan berbagai macam platform teknologi. Untuk *Service interface layer* dibagi menjadi beberapa layer abstraksi : *application service layer,*

business service layer, dan *orchestration service layer* [20].



Gambar 2.1 Enkapsulasi Proses Bisnis dengan *Services* [20]

Secara konseptual pendekatan SOA berdasarkan pada bentuk arsitektur yang merupakan interaksi antara bagian-bagian utama yang meliputi *Service Provider*, *Service Consumer*, *Service Registry*, dan *Service Broker* [21]. SOA menggunakan paradigma *find-bind-execute*, dimana *Service Provider* melakukan *register service*-nya ke dalam *registry public*. Untuk kemudian registry ini digunakan oleh pemakai untuk menemukan service yang sesuai dengan kriteria yang diinginkan. Apabila di dalam *registry* ini terdapat *service* yang dikehendaki, maka pemakai akan diberi kontrak dan alamat akhir *service* tersebut [21].

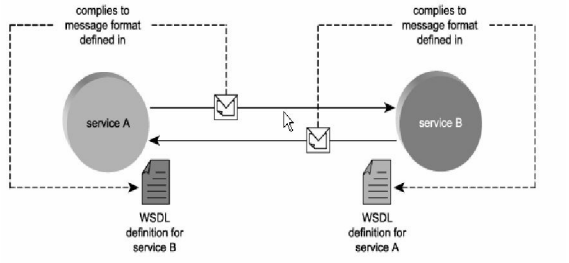


Gambar 2.2 Implementasi *Layer* pada *Enterprise* [20]

2.2 SOA dan *Web Services*

Arsitektur IT yang dikembangkan dengan heterogenitas platform pengembangan memiliki tantangan di dalam proses integrasi aplikasi dan data. Interoperability antar aplikasi harus memberikan jaminan bahwa integrasi bisnis proses secara lebih efektif tanpa mempengaruhi keberadaan aplikasi-aplikasi yang sudah ada sebelumnya.

Teknologi terdistribusi yang sudah ada sebelumnya, seperti CORBA, COM/DCOM, EJB, JAVA RMI, dan lainnya dapat digunakan untuk mengimplementasikan interoperability antar aplikasi, tetapi harus dalam lingkungan platform pengembangan yang sama untuk masing-masing aplikasi [13]. Untuk melakukan interoperability pada *platform* yang heterogen dapat digunakan pendekatan SOA yang berbasiskan pada teknologi web services. Teknologi web *services* memiliki standart terbuka, sehingga dapat memungkinkan aplikasi web *services* yang diimplementasi oleh platform/vendor berbeda dapat berkomunikasi antara satu dengan yang lainnya [23]. Pada dasarnya sebuah *service* di dalam SOA adalah sebuah aplikasi. Aplikasi ini merepresentasikan sebuah *business logic* (*automation logic*) dari sebuah proses sistem besar yang dilingkupinya, harus dapat berdiri sendiri dan berkomunikasi antara satu dengan lainnya [23]. Gambar 3 mendeskripsikan mengenai komunikasi antara satu *service* dengan *service* lainnya yang didefinisikan dengan menggunakan *Web Service Description Language* (WSDL). WSDL mendeskripsikan format sebuah pesan yang akan dikirim oleh sebuah aplikasi web *services*, untuk dapat dimengerti oleh aplikasi web *services* lainnya yang menerima dengan menggunakan teknologi *Simple Object Access Protocol* (SOAP) yang dijalankan pada *Hyper Text Transfer Protocol* (HTTP) [20].



Gambar 2.3 Services Communication [20]

Web services merupakan teknologi yang digunakan untuk mempertukarkan informasi dengan berbasis pada XML, sehingga memungkinkan untuk dapat melakukan interoperability antara beberapa aplikasi yang dikembangkan dengan platform dan bahasa pemrograman berbeda-beda. Pendekatan SOA memungkinkan aplikasi dapat berkomunikasi dengan aplikasi yang lainnya dengan cara meminta *service* dari aplikasi yang bersangkutan tanpa memperhatikan platform dan bahasa pemrograman. Aplikasi dapat beradaptasi terhadap fungsi dan *service* untuk menyesuaikan dengan proses binis yang berbeda secara fleksibel [13].

2.3 *Business Process Execution Language* (BPEL)

BPEL adalah bahasa berbasis XML yang digunakan untuk mendefinisikan proses bisnis dengan web *services*. Tujuan utama dari BPEL adalah untuk standarisasi aliran proses bisnis yang didefinisikan untuk dapat bekerjasama dengan menggunakan web *services* [24]. BPEL memperluas model interaksi web *services* dan memungkinkan untuk mendukung transaksi. BPEL berbasis pada web *services*, sehingga diasumsikan bahwa masing-masing proses bisnis yang terlibat diimplementasikan dengan menggunakan web *services*, sehingga BPEL dapat mengatur interaksi yang terjadi antar web *services* dengan menggunakan dokumen XML [25]. BPEL digunakan untuk mendeskripsikan suatu bisnis proses dengan dua cara yang berbeda, yaitu dengan cara *executable* dan *abstract processes*. Spesifikasi *executable processes* adalah menentukan urutan eksekusi dari sejumlah aktivitas, mitra yang telibat, mempertukarkan pesan antar mitra, dan mekanisme *exception handling*. Spesifikasi *abstract processes* adalah perilaku pertukaran pesan dengan pihak yang berbeda tanpa memberikan informasi mengenai perilaku internal dari mereka. Elemen-elemen yang terdapat pada struktur umum BPEL adalah terdiri dari tag : *Process, PartnerLinks, Variables, dan Sequence* [26]. *Process* merupakan elemen utama dari BPEL. Nama proses didefinisikan pada *tag* proses dengan atribut name. Selain itu tag proses digunakan juga untuk memasukkan informasi yang terkait mengenai definisi proses. PartnerLinks merupakan elemen untuk mendefinisikan tipe port dari service lain yang ikut berpartisipasi dalam eksekusi dari proses bisnis. Variables merupakan elemen yang digunakan untuk menyimpan informasi status yang digunakan selama *workflow logic* berjalan. *Sequence* merupakan elemen untuk mengorganisasikan sekumpulan aktivitas sehingga dapat dieksekusi dalam keterurutan. Adapun elemen-elemen yang didukung BPEL untuk *sequence* diantaranya adalah *recieve, assign, invoke,* dan *reply*.

2.4 Konsep Interoperabilitas Multi *Platform*

Pada bagian ini akan diberikan mengenai model secara konseptual mengenai *framework interoperability* pada e-Gov Indonesia yang digunakan untuk *interoperability* dari pemerintah daerah ke pemerintah pusat atau sebaliknya. Studi kasus yang digunakan adalah pada aplikasi Sistem Informasi dan Administrasi Kependudukan (SIAK) yang dikembangkan oleh Direktorat Jenderal Administrasi Kependudukan Departemen Dalam Negeri (Ditjen Adminduk Depdagri).

1. Interopabilitas E-Gov Sebelumnya

Aplikasi SIAK yang dijalankan di tingkat kecamatan, kabupaten/kota dan provinsi terdiri dari 2 versi, yaitu SIAK online dan offline [5]. SIAK offline dipasang pada setiap kabupaten/kota (aplikasi yang

memiliki fungsi untuk perekam data (pendaftaran penduduk dan pencatatan sipil) dan pencetak data penduduk), dan propinsi (aplikasi yang memiliki fungsi untuk rekap laporan data kependudukan), dimana masing-masing memiliki aplikasi dan *database* sendiri-sendiri. Untuk SIAK *offline* proses interoperabilitas data dilakukan dengan cara mempertukarkan media penyimpanan data secara fisik, seperti disket, CD, dan media penyimpanan yang lain. SIAK online dikembangkan dengan arsitektur aplikasi yang terpusat, sehingga data dan aplikasi hanya berada pada satu tempat, yaitu Ditjen Adminduk Depdagri. SIAK *online* rencananya akan diakses pada tingkat kecamatan (aplikasi yang memiliki fungsi untuk perekam data (pendaftaran penduduk dan pencatatan sipil) dan pencetak data penduduk) yang langsung terkoneksi dengan Ditjen Adminduk Depdagri. Untuk infrastruktur komunikasi SIAK *online* adalah menggunakan VPN *Dial* dengan koneksi secara *synchronous* dari kecamatan ke data center Ditjen Adminduk Depdagri. Untuk SIAK *online* proses interoperabilitas dilakukan secara langsung karena dikembangkan dengan arsitektur terpusat [5].

2. Interoperabiltas E-Gov Yang Dikembangakan

SIAK offline dan online memiliki keterbatasan di dalam hal *interoperability* baik dari sisi aplikasi dan data. SIAK online dengan model koneksi VPN *Dial* harus dapat memberikan jaminan bahwa koneksi dari kecamatan ke data center Ditjen Adminduk Depdagri tidak putus, karena pada saat terjadi koneksi terputus SIAK tidak dapat dioperasikan. Jumlah kecamatan ada sebanyak 5263 jika dilakukan akses secara bersama-sama, maka akan membutuhkan jumlah bandwith yang cukup besar. Sementara infrastruktur IT sampai pada tingkat kecamatan belum semua baik untuk keseluruhan daerah-daerah di Indonesia. SIAK *offline* dan *online* dikembangkan dengan menggunakan pendekatan arsitektur *client-server*.

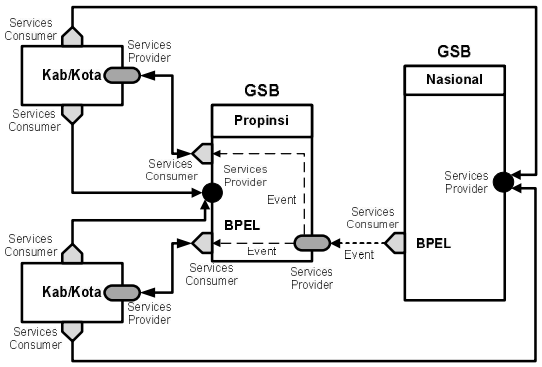
Berdasarkan SIAK yang sekarang ini berjalan, maka perlu dilakukan perbaikan dari sisi arsitekturnya untuk dapat menghasilkan interoperabilitas aplikasi dan data secara lebih baik. Arsitektur interoperability yang dikembangkan adalah menggunakan pendekatan SOA dengan membuat services terhadap sebuah proses bisnis, sehingga *interoperability* dilakukan dengan cara melakukan akses terhadap *services* tersebut. Untuk membuat arsitektur *interoperability* yang bersifat *real time*, maka pendekatan SOA dikombinasikan dengan pendekatan EDA. Kombinasi ini memiliki tujuan agar satu *service* dapat dijalankan dengan adanya *event* sebagai *trigger*, dimana *event* yang terdapat pada *services* didefinisikan dengan WSDL. Untuk mengatur orkestrasi yang terjadi antar *service* maka dibutuhkan sebuah *service bus* yang disebut *Government Service Bus* (GSB) yang didefinisikan dengan BPEL. Arsitektur interoperabilitas yang dikembangkan terdiri dari komponen-komponen seperti pada Gambar 4 :

1. *Client* (*Services Consumer*)
   Komponen ini digunakan untuk melakukan akses terhadap *services* yang disediakan oleh *service provider*.
2. *Services Provider*
   Komponen ini digunakan untuk mendefinisikan bisnis proses ke dalam sebuah *services* dengan menggunakan WSDL.
3. *Event*
   Komponen ini digunakan untuk melakukan trigger terhadap *services* yang didefinisikan dengan menggunakan WSDL.
4. BPEL
   Komponen ini digunakan untuk mengatur orkestrasi antar *services* berdasarkan *trigger* yang diberikan oleh *event.*
5. GSB
   Komponen ini digunakan untuk melakukan deteksi, trigger, dan mendistribusikan *event* (*event*

services). Selain itu komponen ini juga berperan untuk memberikan jaminan bahwa protokol yang berbeda dari platform heterogen dapat berkomunikasi, sekaligus bertanggung jawab untuk mentransformasikan isi pesan (mengganti atau menambahkan pesan) keseluruh *services* yang berpartisipasi (mediasi *service*).

Interoperabilitas yang dikembangkan menggunakan skenario pendekatan bahwa untuk setiap kabupaten/kota, provinsi, dan nasional adalah diasumsikan sebagai elemen-elemen yang terlibat di dalam mekanisme interoperabilitas. Untuk setiap elemen tersebut dapat berperan sebagai *service provider* dan sekaligus sebagai *service consumer*. Arsitektur interoperability yang dikembangkan menggunakan model skenario seperti yang diberikan pada Gambar 4 adalah sebagai berikut :

1. Nasional dan provinsi sebagai *service provider* untuk fungsi tambah, hapus, dan rubah yang akan digunakan oleh *service consumer* kabupaten/kota. Skenario ini digunakan untuk membuat basisdata secara terdistribusi yang terdapat pada setiap kabupaten/kota, provinsi, dan nasional.
2. Kabupaten/kota sebagai *service provider* untuk fungsi *read* data (data penduduk, data kelahiran, data kematian, rekap jumlah penduduk, rekap jumlah kelahiran, dan rekap jumlah kematian) yang akan digunakan oleh *service consumer*, yaitu provinsi dan nasional sesuai dengan struktur hirarki yang dimiliki.
3. Untuk setiap provinsi dan nasional memiliki GSB yang dilengkapi dengan BPEL untuk mengatur orkestrasi yang terjadi antar *services*. Pada GSB juga didefinisikan *event* (proses *read* data) yang digunakan untuk melakukan *trigger* antar *services* yang dilakukan orkestrasi oleh BPEL. Skenario ini berdasarkan bahwa kondisi di Indonesia memiliki 34 provinsi, dan masing-masing provinsi memiliki beberapa kabupaten/kota, sehingga mekanisme interoperabilitasnya membutuhkan orkestrasi antar *services* yang dimiliki oleh masing-masing elemen yang terlibat.



Gambar 2.4 Konseptual Interoperabilitas Multi *Platform*

2.5 Implementation Interoperabilitas Multi *Platform*

Untuk mengukur kebenaran model arsitektur interoperability secara konseptual, maka dari model konseptual tersebut kemudian dilakukan implementasi sebagai simulasi untuk menguji kebenarannya model konseptual. Adapun untuk melakukan simulasi, platform yang digunakan pada kabupaten/kota, provinsi, dan nasional adalah menggunakan platform pengembangan yang bervariasi, seperti yang diberikan pada Tabel 1.

Tabel 2.1 *Platform* Pengembangan (skala eksperimen)

Elemen	Bahasa Program	Database
Kabupaten/Kota	Php	MySQL
Provinsi	Java	Postgre SQL
Nasional	Java	SQL Server

Services yang digunakan didefinisikan berdasarkan *services consumer* dan *services*

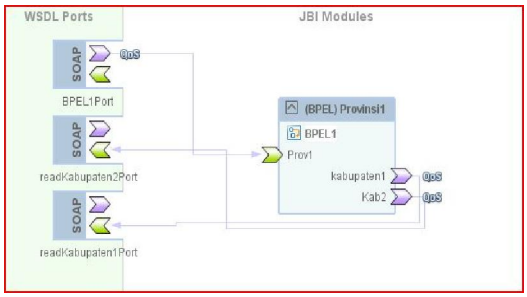
provider yang akan ditempatkan pada masing-masing elemen, seperti yang diberikan pada Tabel 2.

Tabel 2.2 Pendefinisian *Services*

Elemen	Services Consumer	Services Provider
Kabupaten/Kota	Fungsi tambah, hapus, dan rubah	Fungsi *read* yang meliputi : Data penduduk, data kelahiran, data kematian, rekap jumlah penduduk, rekap jumlah kelahiran, dan rekap jumlah kematian
Provinsi	Fungsi *read* yang meliputi : Data penduduk, data kelahiran, data kematian, rekap jumlah penduduk, rekap jumlah kelahiran, dan rekap jumlah kematian	Fungsi tambah, hapus, dan rubah
Nasional	Fungsi *read* yang meliputi : Data penduduk, data kelahiran, data kematian, rekap jumlah penduduk, rekap jumlah kelahiran, dan rekap jumlah kematian	Fungsi tambah, hapus, dan rubah

Services tambah, hapus, dan rubah yang ditempatkan pada provinsi dan nasional adalah menggunakan skenario hubungan satu-ke-satu tanpa menggunakan BPEL, yaitu cukup menggunakan WSDL yang di akses langsung oleh kabupaten/kota. Skenario ini bertujuan untuk membuat basisdata terdistribusi pada masing-masing provinsi dan nasional, dengan mekanisme yang dilakukan secara bersama-sama, yaitu pada saat kabupaten/kota melakukan tambah data penduduk maka provinsi dan nasional juga akan ditambahkan data penduduknya. Kemudian, untuk *services read* yang ditempatkan pada kabupaten/kota dan provinsi adalah menggunakan skenario satu-ke-banyak dengan menggunakan *event* sebagai *trigger* dan BPEL untuk mengatur orkestrasi dengan *services* lain sebagai *partner*. Untuk dapat melakukan deteksi *event* dan orkestrasi antar *services*, maka dibutuhkan GSB, dimana GSB yang digunakan adalah menggunakan *GlassFish* 3.1.2 server sebagai *open* ESB. Mekanisme untuk pertukaran *message* adalah menggunakan model *In-Out*, yang merupakan model pertukaran *message* secara dua arah untuk memenuhi pertukaran *message* yang bersifat asinkron. Arsitektur ini memiliki *service consumer* dan *service provider*, serta *services* unit yang lain secara berbeda-beda termasuk orkestrasi yang

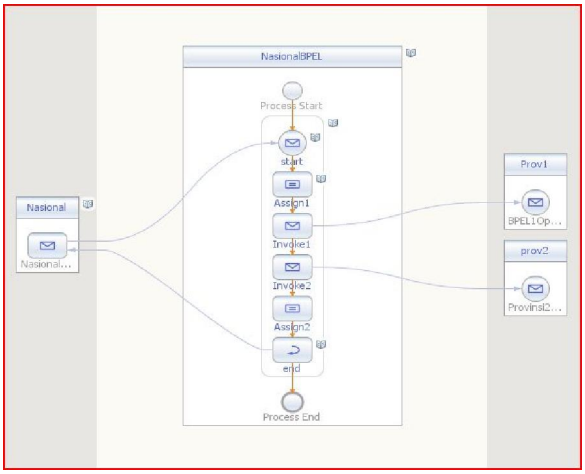
terjadi, maka dibutuhkan untuk di paketkan menjadi sebuah paket gabungan yang disebut dengan *service assembly* seperti pada Gambar 2.5



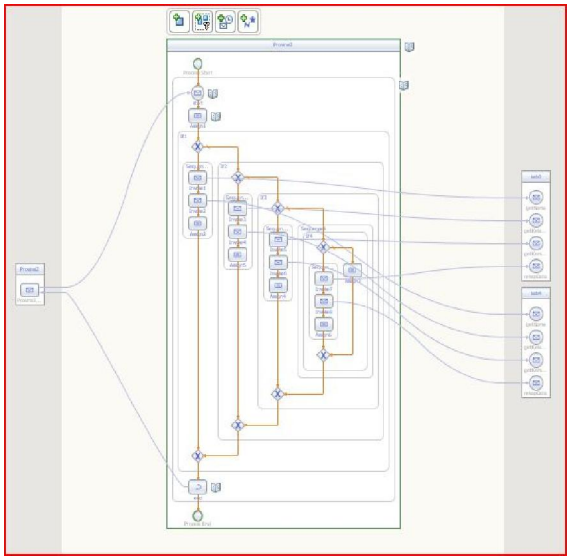
Gambar 2.5 *Service Assemble*

Pada Gambar 2.5, *Java Business Integration* (JBI) modul terdiri dari beberapa komponen yang saling berinteraksi dengan menggunakan model *services*. Untuk komponen yang melakukan *supply* atau konsumsi *service* dilakukan oleh BPEL *Service Engine* yang menyediakan *service* untuk melakukan eksekusi bisnis proses. Bisnis proses yang dieksekusi didefinisikan dengan menggunakan BPEL yang merupakan bahasa berbasis XML. Eksekusi bisnis proses meliputi pertukaran atau orkestrasi *message* antara bisnis proses dengan web *services* lainnya sebagai *partner*. Kontrak antara bisnis proses dengan *service* sebagai *partner* didefinisikan dengan WSDL. *Message* yang dipertukarkan antara bisnis proses dengan *service partner* di *wrapped* ke dalam WSDL sesuai dengan spesifikasi pada JBI, yang selanjutnya oleh JBI akan diarahkan ke *Normalized Message Router* (NMR). NMR berinteraksi dengan web *service* eksternal melalui komponen *binding*. Komponen binding ini bertanggung jawab untuk membungkus detil rician *protocol* pertukaran *message*.

Pada Gambar 2.6 nasional melakukan *request* yang terdiri dari *event* dan parameter data, kemudian dilakukan *assign* dengan cara melakukan *copy event* dan parameter data tersebut untuk digunakan pada *invoke* sebagai parameter *input*. *Event* dan parameter data ini memberikan notifikasi ke provinsi untuk digunakan sebagai *request* ke kabupaten/kota berdasarkan *event* yang di *request* oleh nasional. BPEL provinsi pada Gambar 2.7, *event* dan parameter *request* yang diterima sebagai notifikasi dari nasional, juga dilakukan *assign* dan *invoke* kembali. Selanjutnya, berdasarkan *event* yang sesuai pada s*equence* BPEL provinsi dilakukan *request service* ke kabupaten/kota. Kabupaten/kota memberikan response yang di *assign* sesuai parameter *response* pada *invoke* BPEL provinsi, dengan cara melakukan *concat* parameter data dari sejumlah kabupaten/kota. Kemudian provinsi juga memberikan *response* yang di *assign* sesuai parameter *response* pada *invoke* BPEL nasional, dengan cara melakukan concat parameter data dari sejumlah provinsi. Hasil yang diberikan sebagai response ke nasional adalah dalam bentuk parameter data yang diperoleh dari data-data yang berada di kabupaten/kota sesuai dengan *event* yang diberikan nasional sebagai *request*.



Gambar 2.6 Nasional BPEL



Gambar 2.7 Provinsi BPEL

2.6 Pengukuran Kinerja

Untuk melakukan evaluasi arsitektur interoperabilitas yang sudah dihasilkan, maka dilakukan pengukuran terhadap lamanya waktu eksekusi dengan sejumlah data yang bervariasi pada semua model *services* yang didefinisikan. Lingkungan yang digunakan untuk melakukan evaluasi lamanya waktu eksekusi adalah menggunakan prosesor Intel Core 2 Duo 2.40 GHz dengan *memory* sebesar 2 MB dan *harddisk* sebesar 100 GB. Evaluasi ini dilakukan dengan cara mengukur lamanya waktu eksekusi untuk akses dengan menggunakan SOA+EDA (koneksinya tidak langsung ke *database* kabupaten/kota, tetapi dengan menggunakan services yang dilakukan orkestrasi), Tanpa SOA+EDA (koneksi langsung ke database kabupaten/kota), dan LOCAL (koneksi langsung ke *database* lokal nasional yang diperoleh dari *service* tambah, hapus, dan rubah). Evaluasi ini bertujuan untuk mengetahui tingkat kelayakan model *services* yang didefinisikan pada EAF ini dengan cara melakukan perbandingan lamanya waktu eksekusi antara dengan BPEL, tanpa BPEL, dan LOCAL. Tabel 3 menyajikan hasil pengukuran lamanya waktu eksekusi untuk menampilkan rekap jumlah data penduduk per provinsi, kabupaten, kecamatan, dan kelurahan. Tabel 4 menyajikan hasil pengukuran lamanya waktu eksekusi untuk menampilkan keseluruhan data penduduk.

Tabel 2.3 Waktu Eksekusi Rekap Jumlah Penduduk Nasional

Jumlah Data (record)	Rekap Per	Tanpa SOA+EDA (sec)	SOA+EDA (sec)	LOCAL (sec)
6500	Propinsi	7.833	0.492	0.201
	Kabupaten	5.975	0.112	0.067
	Kecamatan	6.349	0.227	0.079
	Kelurahan	5.842	0.218	0.082
13000	Propinsi	13.571	0.765	0.281
	Kabupaten	10.662	0.187	0.124
	Kecamatan	11.992	0.374	0.202
	Kelurahan	11.721	0.312	0.202
19500	Propinsi	20.443	1.153	0.369
	Kabupaten	17.876	0.211	0.164
	Kecamatan	18.781	0.399	0.292
	Kelurahan	17.692	0.335	0.287
26000	Provinsi	25.993	1.683	0.499
	Kabupaten	23.723	0.322	0.183
	Kecamatan	22.918	0.413	0.399
	Kelurahan	21.769	0.467	0.401
32500	Provinsi	33.842	2.118	0.572
	Kabupaten	31.384	0.488	0.231
	Kecamatan	31.542	0.516	0.421
	Kelurahan	32.738	0.534	0.437
39000	Provinsi	41.883	2.668	0.722
	Kabupaten	38.667	0.589	0.301
	Kecamatan	38.994	0.692	0.481
	Kelurahan	38.478	0.671	0.492
45500	Provinsi	53.881	3.276	0.871
	Kabupaten	50.773	0.608	0.316
	Kecamatan	51.692	0.721	0.502
	Kelurahan	51.462	0.716	0.512

Waktu eksekusi untuk yang menggunakan BPEL pada Tabel 3 dan 4 adalah tidak menggunakan waktu pertama kali dilakukan eksekusi, karena dibutuhkan untuk melakukan kompilasi modul dan *class* terlebih dahulu sehingga waktunya relatif lama. Untuk akses yang selanjutnya tidak dibutuhkan lagi kompilasi modul dan *class* tersebut, sehingga waktunya relatif cepat dan stabil untuk akses-akses berikutnya.

Tabel 2.4 Waktu Eksekusi Menampilkan Data Penduduk Nasional

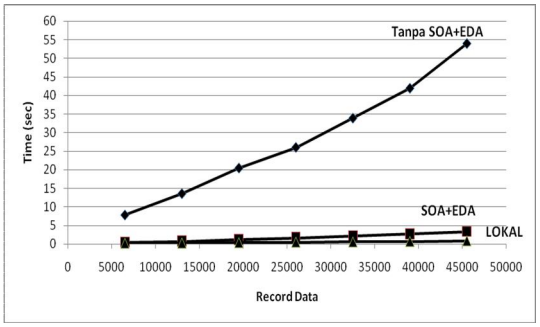
Jumlah Data (record)	Tanpa BPEL (sec)	BPEL (sec)	LOCAL (sec)
6500	25.762	17.354	1.742
13000	43.741	32.479	2.344
19500	55.822	47.199	3.988
26000	70.553	58.831	5.736
32500	82.439	70.773	7.271
39000	97.761	86.359	8.709
45500	113.992	98.794	10.851

Kemudian dari hasil eksperimen pada Tabel 2.3 dan Gambar 2.8. dilakukan uji statistik dengan model *oneway anova* diperoleh hasil uji yang diberikan pada Tabel 2.5. Dari hasil uji statistik dapat diperoleh bahwa nilai siq=0,967 (>0,05) perbedaan waktu eksekusi antara SOA+EDA dengan LOKAL, sehingga dapat disimpulkan bahwa tidak terjadi perbedaan secara signifikan. Hal ini dikarenakan response yang diberikan oleh model *service* ini dalam bentuk output parameter data tunggal. Kemudian, perbandingan antara SOA+EDA dengan tanpa SOA+EDA terjadi perbedaan sangat signifikan karena nilai sig=0 (<0,05). Hal ini dikarenakan koneksi yang dilakukan adalah akses secara langsung ke *database* masing-masing kabupaten/kota, sehingga membutuhkan konsumsi *resources* lebih banyak.

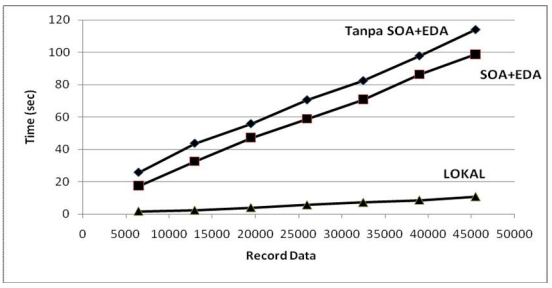
Tabel 2.5 Hasil Uji Statistik Oneway Anova (Rekap Jumlah Penduduk Nasional)

(I) jenis	(J) jenis	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Tanpa SOA + EDA	SOA + EDA	26.47014*	5.00635	.000	13.6931	39.2472
	LOKAL	27.70443*	5.00635	.000	14.9274	40.4814
SOA + EDA	Tanpa SOA + EDA	-26.47014*	5.00635	.000	-39.2472	-13.6931
	LOKAL	1.23429	5.00635	.967	-11.5427	14.0113
LOKAL	Tanpa SOA + EDA	-27.70443*	5.00635	.000	-40.4814	-14.9274
	SOA + EDA	-1.23429	5.00635	.967	-14.0113	11.5427

Kemudian dari hasil eksperimen pada Tabel 2.4 dan Gambar 2.9 dilakukan uji statistik dengan model *oneway anova* diperoleh hasil uji yang diberikan pada Tabel 2.6. Dari hasil uji statistik dapat diperoleh bahwa nilai siq=0,036 (<0,05) perbedaan waktu eksekusi antara SOA+EDA dengan LOKAL, sehingga dapat disimpulkan bahwa terjadi perbedaan secara signifikan. Hal ini dikarenakan response yang diberikan oleh model *service* ini dalam bentuk *output* parameter tidak tunggal yang merupakan kumpulan *array* dari keseluruhan record data sehingga membutuhkan resource yang lebih banyak. Perbandingan antara SOA+EDA dan tanpa SOA+EDA sama-sama membutuhkan *resource* yang banyak. Untuk SOA+EDA lebih cepat karena tidak melakukan koneksi secara langsung ke *database*, tetapi aksesnya menggunakan *services* yang berbasis dokumen teks dengan format XML.



Gambar 2.8 Waktu Eksekusi Rekap Jumlah Penduduk Nasional



Gambar 2.9 Waktu Eksekusi Menampilkan Data Penduduk Nasional

Pada Gambar 2.9 perbedaan waktu eksekusi antara BPEL dengan LOCAL terjadi perbedaan sangat signifikan, karena response yang diberikan oleh model *service* ini dalam bentuk *output* parameter tidak tunggal yang merupakan kumpulan *array* dari keseluruhan *record* data sehingga membutuhkan *resource* yang lebih banyak. Perbandingan antara BPEL dan tanpa BPEL sama-sama membutuhkan *resource* yang banyak. Untuk BPEL lebih cepat karena

tidak melakukan koneksi secara langsung ke *database*, tetapi aksesnya menggunakan *services* yang berbasis dokumen *text* dengan format XML.

Tabel 2.6 Hasil Uji Statistik Oneway Anova (Menampilkan Data Penduduk Nasional)

(I) jenis	(J) jenis	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Tanpa SOA + EDA	SOA + EDA	75.11557	22.71080	.010	17.1539	133.0772
	LOKAL	128.13671	22.71080	.000	70.1751	186.0984
SOA + EDA	Tanpa SOA + EDA	-75.11557	22.71080	.010	-133.0772	-17.1539
	LOKAL	53.02114	22.71080	.036	-4.9405	110.9828
LOKAL	Tanpa SOA + EDA	-128.13671	22.71080	.000	-186.0984	-70.1751
	SOA + EDA	-53.02114	22.71080	.036	-110.9828	4.9405

3. PENUTUP

Berdasarkan hasil eksperimen dengan cara melakukan pengukuran lamanya waktu eksekusi model *services* pada Tabel 3 dan 4, maka arsitektur interoperabilitas multi *platform* ini dapat dinyatakan memenuhi standard kelayakan karena hasil kinerja yang diperoleh (SOA+EDA) jika dibandingkan dengan LOKAL tidak memiliki perbedaan yang signifikan berdasarkan uji statistik yang telah dilakukan khususnya pada bagian model *service* dengan yang memiliki *response* parameter tunggal. Kemudian pada model *services* dengan *response* parameter tidak tunggal berdasarkan hasil pengukuran (SOA+EDA) masih memiliki perbedaan yang sangat signifikan jika dibandingkan dengan LOKAL, walaupun masih lebih baik jika bandingkan dengan Tanpa SOA+EDA.

Oleh karena itu berdasarkan hasil eksperimen ini, pada penelitian berikutnya akan dilakukan perbaikan kinerja arsitektur interoperabilitas multi *platform* ini, khususnya pada bagian model *services* yang memiliki *response* parameter *output* tidak tunggal dengan cara melakukan perubahan skenario interaksi antar *services* untuk dapat menghasilkan kinerja yang lebih baik.

4. DAFTAR PUSTAKA

[1] Patricia, J., Pascual, (2003), *e-Government*, e-Asean Task Force UNDP- APDIP.

[2] Hazlett, S., A., and Hill, F., (2003), E-government: the realities of using IT to transform the public sector, Managing Service Quality, *EJEG*, 13(6): 445-452.

[3] Sosiawan, E.,A., (2008), Tantangan Dan Hambatan Dalam Implementasi E-Government Di Indonesia, *Prosiding Seminar Nasional e-Indonesia*, ITB, Bandung, 2008, 46-59.

[4] Hasibuan, Z. A., (2007), Langkah-langkah Strategis dan Taktis Pengembangan E-Government untuk Pemda, *Jurnal Sistem Informasi MTI UI*, 3(1) : 61-70.

[5] Setiadi, H., Hasibuan, Z. A., and Fahmi, H., (2007), Perubahan Arsitektur Database Dan Aplikasi Administrasi Kependudukan Yang Sejalan Dengan Otonomi Daerah, *Jurnal Sistem Informasi MTI UI*, 3(1) : 42-51.

[6] Instruksi Presiden Nomor 3 Tahun 2003 tentang Kebijakan dan Strategi Nasional Pengembangan e-Government, Jakarta, 2003.

[7] Blue Print Sistem Aplikasi E-Government, Departemen Komunikasi Dan Informatika Republik Indonesia, Jakarta, 2004.

[8] Rabaiah, Abdelbaset, 2007, Federation of E-government: A Model and Framework, *Information Systems Audit and Control Association Journal*, 4 : 1-4.

[9] Allen, B.A., Juillet, L. Paquet, P. and Roy, J., (2001), E-Governance & government on-line in Canada : Partnerships, people & prospects, *Government Information Quarterly*, 18 : 93–104.

[10] Janssen, Marijn, and Cresswell, Anthony, (2005), Enterprise Architecture Integration in E-government, *Journal of Enterprise Information Management*, 18 (5) : 531 - 547.

[11] Ioannidis, Anastasios, Spanoudakis, Manos, and Priggouris, Giannis, (2003), An Xml-Based Platform For E-Government Services Deployment, *Journal of Web Engineering*, 1(2) :147-162

[12] Lee, Thomas, Hon , C., T., and Cheung, David, (2009), XML Schema Design and Management for e-Government Data Interoperability, *Electronic Journal of e-Government*, 7(4) : 381 – 390.

[13] Phu, Phung Huu, and Yi, Myeongjae, (2005), A Service Management Framework for SOA-based Interoperability Transactions, *Proceedings The 9th Russian-Korean International Symposium on, Korean*, 680-684.

[14] Contenti, Mariangela, (2005), A Distributed Architecture for Supporting e-Government Cooperative Processes, *TCGOV*, 181–192.

[15] Weerakkody, V., Janssen, M., and Hjort-Madsen, Kristian, (2007), Integration and Enterprise architecture challenges in E-Government: a European perspective, *International Journal of Cases on Electronic Commerce*, 3(2) : 13-36.

[16] Müller, Viktor, Simon, Balázs, László, Zoltán, and Goldschmidt, Balázs, (2009), Business Monitoring Solution for an e-Government Framework based on SOA Architecture, *Proceeding of the 13th IASTED International Conference : Software Engineering and Applications (SEA), Cambridge, MA, USA*, 2-4.

[17] Peristeras, Vassilios, Tarabanis, Konstantinos, and Goudos, Sotirios K., Model-driven eGovernment interoperability: A review of the state of the art, *Computer Standards & Interfaces Journal, Science Direct* 31 : 613–628.

[18] Stefanova, Kamelia, Design Principles of Identity Management Architecture Development for Cross-Border eGovernment Services, (2010), *Journal of e- Government*, 8 (2) : 189-202.

[19] Braun, Christian, and Winter, Robert, (2007), Integration of IT Service Management into Enterprise Architecture, *Proceedings of the 2007 ACM symposium on Applied computing*, ACM New York, NY, USA, 1215 – 1219.

[20] Erl, Thomas, (2005), *Service-Oriented Architecture: Concepts*, Technology, and Design, Prentice Hall PTR, New Jersey.

[21] Open SOA, Service Component Architecture (SCA), http://osoa.org/display/Main/Service+ Component+Architecture+Home, 2007. (Last access: Juli 3, 2009).

[22] Michelson, B.,M., (2006), Event-Driven Architecture Overview, Patricia Seybold Group, OMG, February 2, pp. 1-8.

[23] Sahin, K., (2008), Service Oriented Architecture (SOA) Based Web Services For Geographic Information Systems, T*he International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Beijing, Vol. XXXVII, Part B2, pp. 625-630.

[24] Zhong D., et al., (2007), Reliability Prediction for BPEL-Based Composite Web Service, *Proceedings of The FirstInternational Conference on Research Challenges in Information Science*, Ouarzazate, Morocco, 2007, 265-270.

[25] Hafid, B., Balouki, Y., Laassiri, J., Benaini, R., Bouhadadi, M., and Hajji,

S. E., (2009), Using BPEL for Behavioral Concepts in ODP Computational Language, *Proceedings of the World Congress on Engineering*, London, U.K., 2009, 1 : 123-129.

[26] Juric, M., B., and Pant, K., WSDL and UDDI Extensions for Version Support in Web Services, *Journal of Systems and Software*, 82 : 1326-1343.