BAB 2
BPEL dan ESB

2.1 *Business Process Execution Language* (BPEL)

Di dalam perusahaan, BPEL digunakan untuk standardisasi EAI dan memperluas integrasi sistem yang masih terisolasi. Antar perusahaan, BPEL memungkinkan integrasi dengan partner bisnis. BPEL memungkinkan perusahaan untuk mendefinisikan proses bisnis, memperbaiki proses bisnis sehingga lebih sesuai.

Penerapan suatu proses bisnis memerlukan standard dan bahasa khusus untuk komposisi ke dalam proses bisnis yang menunjukkan proses bisnis secara standard pula, dengan menggunakan bahasa yang dapat diterima umum. BPEL merupakan bahasa yang dapat diterima umum dan cepat menjadi standard yang dominan. Tujuan utama BPEL adalah untuk standardisasi otomasi proses antar WS.

Pada SOA, layanan-layanan menjadi blok bangunan utama dari keseluruhan arsitektur. Layanan juga merupakan blok bangunan utama dalam proses BPEL. Layanan-layanan tersebut merupakan operasi *coarse-graine* yang disebut layanan bisnis. Operasi dalam layanan bisnis biasanya merepresentasikan aktivitas bisnis. Layanan bisnis ini digunakan dalam proses bisnis atau tepatnya BPEL menggunakan layanan bisnis ini untuk

mengeksekusi aktivitas proses. Dengan kata lain, proses pada BPEL merupakan komposisi dari layanan bisnis. Layanan bisnis menyediakan fungsionalitas, sedangkan proses BPEL berisi aliran proses (Juric et al, 2010).

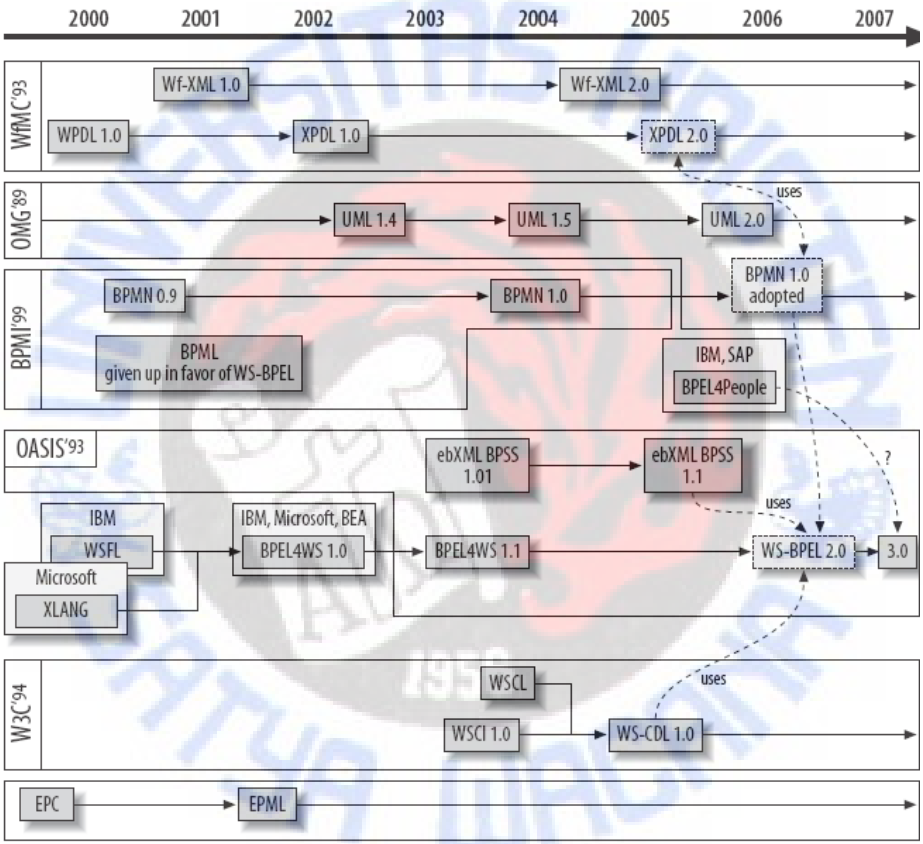
Menurut Louridas (2008), program BPEL mempunyai empat bagian utama yang terdiri dari :

1. Bagian <partnerLinks> mendefinisikan pihak-pihak yang bekerjasama untuk memenuhi proses bisnis.
2. Bagian <variables> yang mendefinisikan struktur data yang akan digunakan program. Definisi mengacu pada jenis pesan WSDL dan elemen dan jenis XSD (XML schema definition)
3. Bagian <faultHandlers>, digunakan untuk menangani program yang akan melakukan invokasi ketika ada kesalahan.
4. Bagian <sequence> mendefinisikan prosedur proses yang dikomposisi dari *BPEL activities*.

2.2 Evolusi standard BPEL

Menurut Juric et al (2010) BPEL 1.0 dikembangkan oleh IBM, BEA dan Microsoft pada Agustus 2002. Kemudian SAP dan Siebel kemudian bergabung yang menghasilkan bebererapa modifikasi dan perbaikan yang muncul di versi 1.1 pada Maret 2003. Pada April 2003, BPEL dimasukkan ke OASIS (*Organization for the Advancement of Structured Information Standards*) untuk tujuan standardisasi, sementara itu WSBPEL TC (*Web services Business Process Execution Language Technical Committee*) telah dibentuk. Sejak itu, banyak *vendor* yang kemudian bergabung ke WSBPEL TC (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=WSBPEL). Hal ini telah

mengakibatkan BPEL dapat diterima secara luas di industri. Pada april 2007, BPEL versi 2.0 dikenalkan oleh OASIS. Evolusi perkembangan standard BPEL dapat dilihat pada Gambar 3.12, sedangkan penjelasan tentang standard manajemen proses bisnis dapat dilihat pada Tabel 3.1.



Gambar 3.12 Evolusi perkembangan standard BPEL (Josuttis, 2007; Piispanen, 2008)

Tabel 3.1 Penjelasan Standard manajemen proses bisnis (Havey, 2005; Piispanen, 2008)

Standard	Organisasi	Deskripsi
Business Process Execution Language (BPEL)	OASIS	Bahasa BPM paling terkenal; menyajikan sebuah proses sebagai XML dengan WS bindings
Business Process Modeling Language (BPML)	Business Process Modeling Initiative (BPMI)	Bahasa proses XML yang mirip dengan BPEL
Business Process Modeling Notation (BPMN)	BPMI	Bahasa grafis dengan memetakan ke BPEL
Workflow Reference Model	Workflow Management Coalition (WfMC)	Pendekatan arsitektur dasar ke workflow/BPM
Workflow API (WAPI)	WfMC	API fungsional dan administrative dengan definisi dalam bahasa C, IDL, dan COM
XML Process Definition Language (XPDL)	WfMC	Bahasa proses XML yang mirip dengan BPEL
Workflow XML (WfXML)	WfMC	Bahasa XML untuk komunikasi berbasis WS antara *workflow runtime engines*
Web services Choreography Antarmuka (WSCI)	World Wide Web Consortium (W3C)	Bahasa XML yang matang untuk koreografi WS atau interaksi berorientasi proses dari WS diantara banyak partisipan

Standard	Organisasi	Deskripsi
Web services Choreography Description Language (WS-CDL)	W3C	Bahasa koreografi XML resmi dari W3C
Web services Conversation Language (WSCL)	W3C	Bahasa koreografi XML dasar yang elegan
Business Process Definition Metamodel (BPDM)	Object Management Group (OMG)	Model untuk bahasa proses BPM yang dibangun menggunakan Model Driven Architecture (MDA)
Business Process Runtime Antarmuka (BPRI)	OMG	Model MDA untuk BPM API fungsional dan administratif
XLANG	Microsoft	Bahasa proses XML awal yang dipengaruhi oleh perancangan BPEL
Web services Flow Language (WSFL)	IBM	Bahasa proses XML awal yang juga dipengaruhi oleh perancangan BPEL
Business Process Specification Schema (BPSS)	OASIS	Bahasa proses untuk kolaborasi business-to-business (B2B)

Di antara cabang utama evolusi standard manajemen proses bisnis dapat dibedakan sebagai berikut (Weske, 2007; Havey, 2005):

- Cabang paling terkenal saat ini adalah BPEL
- Cabang utama lain adalah *Workflow Management Coalition* (WfMC), yang ditemukan tahun 1993 untuk menentukan standard untuk sistem

manajemen workflow. Standard pertama adalah *Workflow Process Definition Language* (WPDL), tetapi standard definisi proses saat ini dalam pengaruh XML dan dinamakan *XML Process Definition Language* (XPDL).

- Standard ketiga yang penting adalah *Business Process Modeling Notation* (BPMN), yang awalnya didefinisikan oleh *Business Process Management Initiation* tetapi sekarang dijalankan oleh *Object Management Group* (OMG).

Selain cabang utama, masih ada standard lain yang juga berkembang yang mencakup (Weske, 2007; Havey, 2005):

- Wf-XML, yang merupakan standard yang disediakan oleh WfMC untuk mendefinisikan bagaimana menginstal definisi proses (yang didefinisikan dengan BPEL dan XPDL) ke dalam sebuah mesin proses.
- UML (*Unified Modeling Language*), yang menyediakan beberapa notasi grafis (khususnya Diagram *Activity*) yang digunakan untuk aliran proses (process flow).
- WS-CDL (*Choreography Definition Language*), merupakan standard untuk menentukan proses bisnis melalui koreografi.
- BPSS (*Business Process Specification Schema*), yang merupakan bagian dari standard ebXML (spesifikasi XML untuk *electronic business*)
- EPC (*Event-driven Process Chain*), yang sekarang didukung oleh format pertukaran *EPC Markup Language* (EPML). Format ini (lebih banyak di Eropa) digunakan dengan SAP/R3 dan ARIS.

Hanya BPEL dan XPDL yang disediakan untuk mesin, dan BPEL saat ini tidak mempunyai dukungan notasi (sering disebut "*business process*

assembler"). Untuk alasan ini perusahaan-perusahaan besar bekerja keras agar dapat mentransfer bahasa pemodelan lain dan notasi ke dalam scenario sederhana, karena mentransfer ke scenario yang lebih rumit dari satu model ke model lain menjadi sangat suit. Namun ini lebih sulit dari kedengarannya. Ingat bahwa OMG akhir-akhir ini telah memulai bekerja pada standard untuk mentransfer Model Proses Bisnis. Standard ini dinamakan *Business Process Definition Metamodel* (BPDM)

Notasi paling penting adalah BPMN (lihat Gambar 3.6), UML, dan EPC. Tidak seperti BPEL, yang merupakan format XML murni untuk menentukan proses bisnis, notasi grafis ini merupakan jaminan bahwa Diagram Proses Bisnis akan kelihatan sama, sehingga tidak ada jawabn standar yang akan digunakan.

BPEL merupakan momentum dalam komunitas SOA, dan khusus dalam dunia teknik WS. Namun untuk masyarakat bisnis, BPMN dan EPC nampak lebih intuitif, dan yang mendukung BPMN secara penuh adalah kecanggihan XPDL.

Seperti telah dijelaskan sebelumnya, teknologi WS-* memberikan peran terhadap berkembangnya SOAD, hingga menjadi *Contemporary SOA*. Salah satu permasalahan pada SOA yang dapat diselesaikan oleh teknologi ini adalah pembuatan proses bisnis, dengan WS-BPEL. Untuk mengkomposisi sekumpulan WS menjadi sebuah *workflow*, dibutuhkan sebuah standar. Hal ini dimungkinkan dengan adanya WS-BPEL atau BPEL4WS. Ekstensi WS ini memiliki bahasa yang bisa dikompilasi dan dijalankan oleh aplikasi yang mendukung orkestrasi. Ekstensi ini membawa WS ke dalam integrasi perusahaan. WS-BPEL adalah ekstensi WS yang digunakan untuk

memfasilitasi proses pemodelan dan eksekusi BPEL dalam WS. BPEL sendiri merupakan sebuah bahasa pemodelan berformat XML yang digunakan untuk mendeskripsikan proses bisnis. Model yang dihasilkan oleh bahasa ini nantinya dapat dieksekusi oleh mesin BPEL. Dengan demikian menurut Holanda et al (2010) BPEL telah menjadi standard untuk menetapkan dan mengeksekusi spesifikasi alur kerja untuk invokasi komposisi WS.

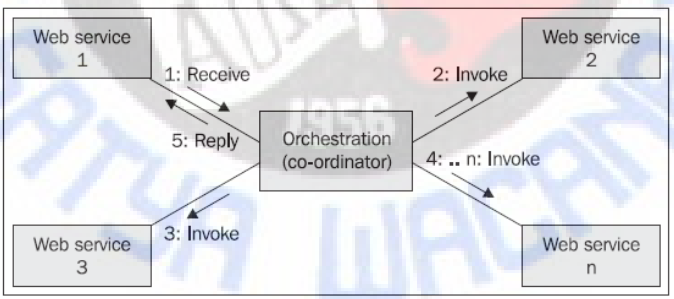
Di dalam perusahaan, BPEL digunakan untuk standardisasi integrasi aplikasi perusahaan dan memperluas integrasi dari sistem sebelumnya yang terisolasi. Diantara perusahaan, BPEL memungkinkan lebih mudah dan lebih efektif mengintegrasikan dengan partner bisnis. BPEL merangsang perusahaan untuk mendefinisikan proses bisnis perusahaan, yang membawa prubahan ke optimalisasi proses bisnis, *reengineering*, dan pemilihan proses yang paling sesuai, sehinga dapat organisasi semakin optimal (Korherr, 2008).

2.3 Orkestrasi dan Koreografi

Tergantung pada kebutuhan, komposisi layanan dapat menekankan pada proses publik atau privat, yang mengacu pada acuan orkestrasi atau koreografi. Pada orkestrasi, proses sentral (yang dapat berupa WS lain) mengendalikan WS yang terlibat dan mengkoordinasi eksekusi operasi WS yang berbeda yang terlibat dalam operasi. Hal ini dikerjakan sesuai dengan kebutuhan dari orkestrasi. WS yang terlibat tidak mengetahui (dan tidak perlu tahu) baha WS tersebut terlibat dalam komposisi dan menjadi bagian dari proses bisnis yang lebih tinggi. Hanya koordinator sentral dari orkestrasi yang mengetahui hal ini, sehingga orkestrasi disentralkan dengan definisi operasi dan urutan invokasi

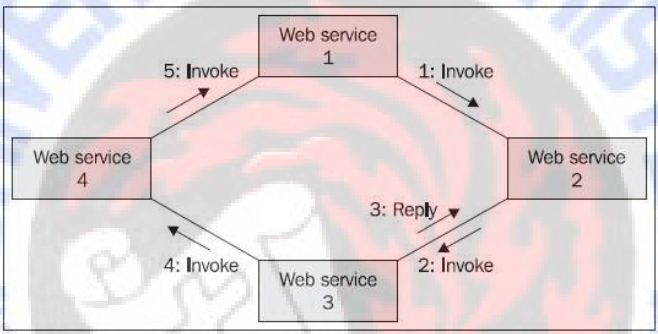
dari WS. Orkestrasi biasanya digunakan untuk proses bisnis privat dan secara skematik dapat dilihat pada Gambar 3.13 dibawah ini (Juric et al, 2010).

Orkestrasi layanan memungkinkan WS dikomposisikan bersama dengan pola yang telah ditetapkan sebelumnya, dideskripsikan menggunakan bahasa orkestrasi dan dieksekusi pada mesin orkestrasi. Orkestrasi dapat berlangsung di banyak aplikasi dan atau organisasi dan menghasilkan proses transaksional yang lama. Layanan sendiri tidak mempunyai pengetahuan mengenai keterlibatannya dalam aplikasi level yang lebih tinggi. Orkestrasi layanan dideskripsikan dari sudut pandang partisipan tunggal (yang dapat berupa WS lain) dan karena itu , proses terpusat akan bertindak sebagai pengendalii terhadap layanan yang terlibat. Bahasa orkestrasi mendeskripsikan interaksi antara WS melalui pengidentifikasian pesan, percabangan lojik, dan rangkaian invokasi. BPEL merupakan bahasa pemodelan proses bisnis yang dapat dieksekusi dan secara de facto menjadi standard cara melakukan orkestrasi WS. BPEL didukung oleh industri secara meluas yang mencakup IBM, Microsoft, dan Oracle dengan implementasi yang konkrit. (Barker et al, 2009)



Gambar 3..13 Orkestrasi WS (Juric et al, 2010)

Koreografi pada sisi lain tidak tergantung pada koordinator sentral. Setiap WS yang terlibat dalam koreografi mengetahui secara pasti kapan mengeksekusi operasi dan dengan siapa berinteriaksi. Koreografi merupakan upaya kolaborasi yang dipusatkan pada pertukaran pesan dalam proses bisnis publik. Semua partisipan dari koreografi sadar dalam proses bisnis, operasi untuk eksekusi, pesan untuk pertukaran, dan waktu pertukaran pesan. Koreografi dalam komposisi WS ditunjukkan dalam Gambar 3.14 berikut ini (Juric et al, 2010).



Gambar 3.14 Koreografi WS (Juric et al, 2010)

2.4 Enterprise Service Bus

ESB merupakan infrastruktur untuk mengintegrasikan aplikasi dan layanan. ESB memperkuat SOA melalui pengurangan jumlah, ukuran, dan kompleksitas antarmuka antara aplikasi dan layanan-layanan. ESB digunakan untuk melakukan koneksi komponen perangkat lunak yang sudah ada dan yang baru untuk membangun sebuah SOA. ESB diperlukan untuk melakukan koneksi ke beberapa sumberdaya TI. ESB harus fleksibel untuk menggabungkan dan memasang ulang komponen sesuai dengan perubahan kebutuhan bisnis. ESB melakukan koneksi komponen yang terikat longgar,

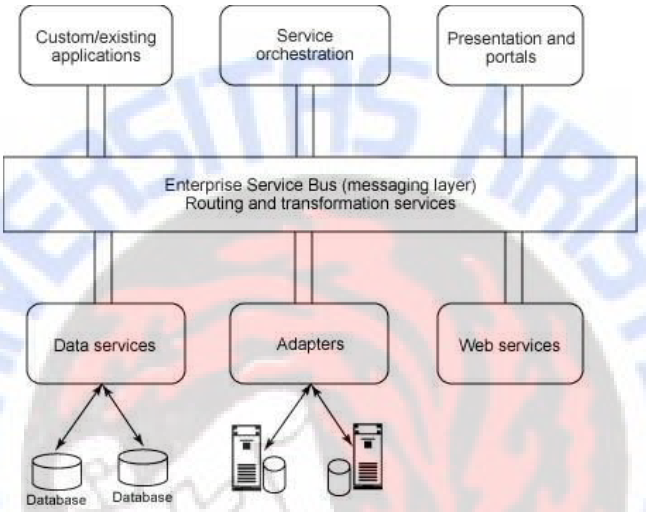
sehingga menyediakan kemampuan untuk mengintegrasikan sistem ke dalam SOA dan men-*deploy* secara bertahap (Juric, 2007; Andary-Sage, 2010).

Pendekatan *services bus* untuk integrasi adalah menggunakan teknologi yang menyediakan *bus* untuk integrasi aplikasi. Aplikasi-aplikasi yang berbeda tidak berkomunikasi satu sama lain secara langsung melainkan berkomunikasi melalui *backbone middleware* SOA. Fitur arsitektur ESB yang paling membedakan adalah sifat terdistribusi dari topologi integrasi. ESB merupakan sekumpulan *middleware* layanan-layanan yang menyediakan kemampuan integrasi. *Middleware* layanan-layanan ini merupakan jantung arsitektur ESB yang menempatkan pesan untuk dapat diroutekan dan ditransformasikan (Juric, 2007; Andary-Sage, 2010).

Arsitektur umum dari ESB dengan komponen yang terkoneksi dapat dilihat pada Gambar 3.15. Komponen dapat mengambil peran penghasil layanan atau pemakai layanan. Layanan-layanan dapat berupa komponen spesial seperti mesin orkestrasi, adapter untuk sumberdaya data atau adapter untuk sistem eksternal dengan transformasi pesan atau konversi transport protokol. ESB melakukan mediasi pesan antar komponen, memutuskan lokasi untuk rute pesan, dan transformasi pesan. ESB memerlukan memori persisten seperti terkoneksi dengan basisdata (Juric, 2007; Andary-Sage, 2010).

Menurut Juric (2007) dan Andary-Sage (2010), satu pendekatan dalam mendefinisikan arsitektur umum ESB adalah spesifikasi Java Business Integration. JBI merupakan standard untuk ESB, sedangkan ESB sendiri merupakan sebuah pola arsitektural untuk SOA. Spesifikasi JBI mendeskripsikan arsitektur *pluggable* bagi kontainer untuk penyedia layanan dan pemakai komponen. Layanan melakukan koneksi melalui *Binding*

Component (BC) atau dapat di-*host* kedalam kontainer sebagai bagian dari *Service Engine* (SE). Layanan-layanan dideskripsikan menggunakan WSDL. Pesan selalu diterjemahkan ke dalam format pesan umum dan dirutekan oleh *Normalized Message Router* (NMR)



Gambar 3.15 Arsitektur ESB secara umum (Juric, 2007; Andary-Sage, 2010)

ESB menyediakan infrastruktur komunikasi antar layanan yang kuat, dapat diandalkan, aman dan dapat diperluas. ESB juga menyediakan kendali komunikasi dan kendali atas penggunaan layanan-layanan yang mencakup (Juric, 2007) :

1. Kemampuan menangkap pesan, yang memungkinkan untuk menangkap pesan *request* untuk layanan-layanan dan pesan *response* dari layanan, serta memberikan pemrosesan tambahan. Dengan cara ini, ESB dapat bertindak sebagai *intermediary.*
2. Kemampuan *routing*, yang memungkinkan ESB melakukan *routing* pesan

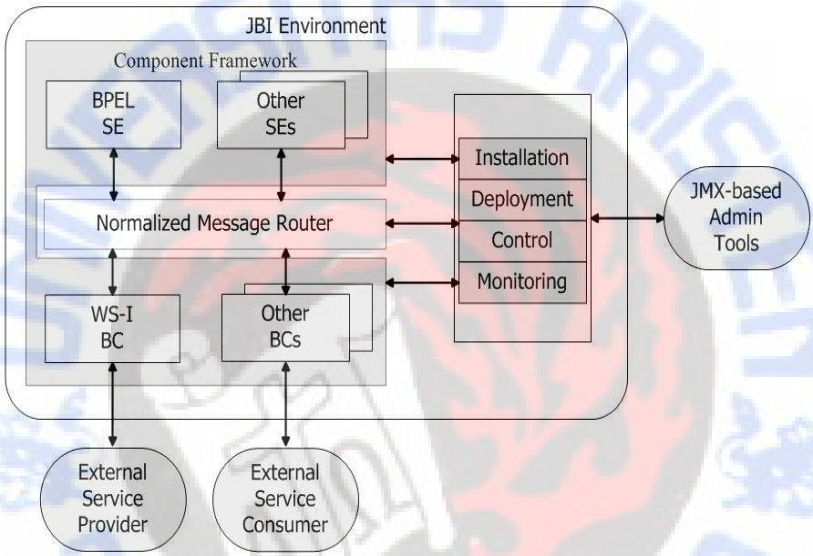
ke layanan-layanan yang berbeda didasarkan pada isi (*content*), asal, atau atribut lain.

3. Kemampuan transformasi, yang memungkinkan transformasi pesan sebelum dikirimkan ke layanan-layanan. Untuk pesan format XML, transformasi semacam ini dilakukan menggunakan XSLT (*Extensible Stylesheet Language for Transformations*) atau mesin XQuery.
4. Kendali atas deployment, penggunaan dan pemeliharaan layanan-layanan. Hal ini memungkinkan adanya *logging*, *profiling*, *load balancing*, *performance tuning*, ongkos penggunaan layanan-layanan, *distributed deployment*, *on-the-fly reconfiguration*, dsb.
5. Fitur manajemen lain yang mencakup definisi korelasi antar pesan, definisi *path* komunikasi yang handal, definisi *security constraints* yang berkaitan dengan pesan dan layanan-layanan, dsb.

2.5 *Java Business Integration*

JBI merupakan sebuah standard yang dikembangkan oleh *Java Community Process* sebagai sebuah pendekatan untuk mengimplementasikan SOA. JBI mendefinisikan lingkungan untuk komponen *plugin* yang berinteraksi menggunakan Model Layanan berbasis WSDL. Komponen *plugin* berfungsi sebagai penyedia layanan atau pemakai layanan atau keduanya. Komponen yang menyediakan atau mengkonsumsi layanan secara lokal (didalam lingkungan JBI) dinamakan *Service Engine*. Komponen yang menyediakan atau mengkonsumsi layanan menggunakan beberapa protokol komunikasi dinamakan *Binding Component*. *Binding Component* dan *Service Engine* hanya berbeda secara logika dan fungsinal, namun secara teknis keduanya mengimplemtasikan antarmuka yang sama. Selain *Service Engine*

dan *Binding Component* masih ada satu komponen lagi yaitu *Normalized Message Router* (NMR). NMR merupakan *backbone* dari JBI yang memudahkan interoperasi antara komponen JBI menggunakan *service descriptor* berbasis WSDL. Pesan dirubah dalam format "*normalized*" (Binildas, 2008, Kumar et al, 2010, Schmutz et al, 2010). Untuk kejelasannya dapat dilihat pada Gambar 3.16.



Gambar 3.16 Arsitektur JBI *Runtime Environment* (Vinoski, 2005; Binildas, 2008; Rosenauer, 2008; Kumar et al, 2010; Schmutz et al, 2010)

Dari Gambar 3.16 dapat dilihat bahwa terdapat tiga komponen dalam JBI *Runtime Environment* yang terdiri dari *Normalized Message Router (*NMR*)*, *Binding Component* (BC) dan *Service Engine* (SE) dengan fungsi masing-masing komponen sebagai berikut (Vinoski, 2005; Binildas, 2008, Kumar et al, 2010, Schmutz et al, 2010):

- *Normalized Message Router* berfungsi membawa pesan dari client dan me-*routing*-kan pesan ke SE yang sesuai untuk pemrosesan dan melewatkan *normalized message* (WSDL) diantara komponen JBI.
- BPEL *Service Engine* melakukan orkestrasi proses bisnis dengan WS-BPEL 2.0.
- XSLT *Service Engine* berfungsi menjalankan transformasi dokumen XML dari satu format ke format lain menggunakan *XSL stylesheets* dan transformasi untuk dideploy sebagai WS yang dapat digunakan oleh client eksternal.
- SQL *Service Engine* berfungsi melakukan eksekusi SQL *Data Definition Language*, SQL *Data Manipulation Language* dan *stored procedures* dari sebuah basisdata.
- IEP *Service Engine* berfungsi memebaca data dari sebuah sumber input dan kemudian memproses ke dalam format yang dapat digunakan untuk berbagai tujuan seperti pelaporan atau informasi business intelegent.
- Java EE *Service Engine* berperan sebagai jembatan antara container JBI yang memungkinkan Java EE WS dikonsumsi dari dalam komponen JBI.
- File *Binding Component* menyediakan mekanisme komunikasi untuk komponen JBI untuk berinteraksi dengan sistem file. File *Binding Component* dapat bertindak sebagai *provider* dengan melakukan pengecekan untuk file baru yang akan diproses, atau sebagai konsumer melalui *outputting* file untuk proses lain atau komponen lain.
- SMTP *Binding Component* menyediakan layanan email ke JBI Server dan bertindak sebagai *provider* dengan menerima pesan SMTP atau bertindak

sebagai konsumer dengan mengirimkan email SMTP ke alamat email eksternal.

- FTP *Binding Component* menyediakan layanan transport FTP ke container JBI yang memungkinkan pesan diterima dan dikirim melalui protokol FTP.
- HTTP *Binding Component* memungkinkan pesan JBI dikirim dan diterima menggunakan SOAP melalui HTTP dan HTTPS. Komponen mendukung RPC Literal, RPC Encoded, dan Document Literal encoding schemes.
- JDBC *Binding Component* untuk berinteraksi dengan basisdata (lebih terbatas dibanding SQL *Service Engine*)
- JMS *Binding Component* memungkinkan container JBI untuk berkomunikasi dengan JMS *message queues* dan *topics*. Komponen dapat bertindak sebagai penyedia atau pemakai dari pesan JMS.