# BAB II

# TINJAUAN PUSTAKA

## A. PENELITIAN TERDAHULU

Integrasi aplikasi telah menjadi tantangan yang serius dalam pembangunan aplikasi yang memerlukan permintaan data dari sumber-sumber data yang heterogen (Al-Sudairy Mohammed T, Vasista T. G. K, 2011). Integrasi telah banyak dilakukan, salah satunya telah dilakukan oleh Georgescu Vasile (2007).

Vasile mengintegrasikan beberapa aplikasi *Data Mining* melalui *portal*. Sistem ini memungkinkan *end user* untuk berkolaborasi dalam pekerjaaannya. Semua fungsionalitas *Data Mining* dapat diakses melalui *web browser* standar. Vasile Membangun sebuah *portal knowledge* pada *layer* paling atas dari sistem *data mining* terdistribusi. *Portal* sebagai sebuah *information gateway* untuk mengirimkan informasi bisnis secara *just-in-time* dan mandiri melalui internet. *Portal* dibangun di atas sebuah sistem bertingkat, yang terdiri dari tiga lapisan: lapisan memori *repository*, lapisan administrasi pengetahuan dan lapisan presentasi.

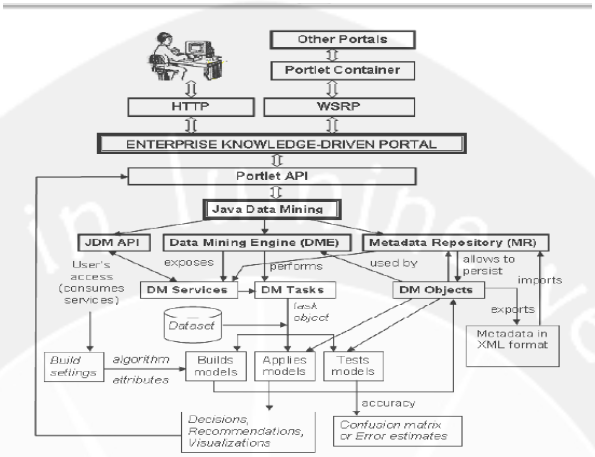
Setiap *data mining* memiliki *Data Mining Engine* (DME), akses terhadap fungsi-fungsi DME dibuat menggunakan *Java Data Mining* (JDM) *Application Programmable Interface* (API). API melindungi pengguna datamining dari implementasi nyata dalam DME dan beberapa komponen yang berhubungan dengan sub komponen yang digunakan oleh DME. Akses terhadap fungsionalitas *Data Mining* dibuat dalam bentuk portlet. Portlet adalah komponen web yang

dapat dipergunakan kembali untuk membangun *portal*. Portlet-portlet tersebut dikelola di dalam sebuah sistem yang disebut dengan *Web Service for Remote Portlets* (WSRP). Spesifikasi WSRP adalah sebuah produk dari *Organization for Advancement of Structured information Standard* (OASIS), dan mendefinisikan sebuah antarmuka *web service* untuk berinteraksi dengan *web service* berorientasi presentasi yang di desain menjadi *plug and play adapter* untuk portal.

WSRP mengijinkan *portal* untuk menampilkan running Portlet secara *remote* di dalam halamannya tanpa menuntut beberapa pemrograman tambahan oleh pembangun portal. Untuk *end user* ini tampak bahwa Portlet berjalan secara lokal di dalam *portal* mereka, tetapi pada kenyataannya portal berada dalam sebuah *container portlet* yang berjalan secara *remote*, dan interaksi terjadi melalui pertukaran pesan *Simple Object Application Protocol* (SOAP).

Poin penting dalam desain sebuah arsitektur berbasis service untuk menyediakan fungsionalitas *data mining* terdistribusi adalah bagaimana konten *web service* diterima sebagai pesan SOAP dapat terlihat bagi *end user*. WSRP standar secara ekplisit ditujukan untuk kebutuhan antar muka *web service*. *Remote* Portlet berjalan sebagai *web service* pada *remote server* yang dipublish dalam direktori UDDI untuk mempermudah *finding* dan *binding*. Proxy Portlet akan meminta layanan WSRP melalui potokol SOAP. Aplikasi menggunakan layanan WSRP dengan membuat *instance remote* Portlet, mengirim *even*t atau meminta *markup* dan yang terakhir menghancurkan *remote* portlet *instance* ketika tidak dibutuhkan. Sebuah arsitektur berorientasi *service* untuk mengeksploitasi

fungsionalitas *Java Data Mining* secara *remote* melalui portal dapat dilihat pada gambar 1 di bawah ini.




Gambar 1. Arsitektur aplikasi Georgescu Vasile (2007).

Tool yang dipergunakan untuk membangun aplikasi berbasis web melalui portal adalah Google Web Toolkit (GWT). GWT menghadirkan sebuah pendekatan yang benar-benar berbeda untuk membangun aplikasi web yang didesain untuk kelihatan sangat mirip dengan aplikasi desktop. GWT merupakan compiler Java ke Java Script yang memproduksi kode berkemampuan berjalan pada *browser* standar seperti Internet Explorer, Firefox, Safari dan Opera. GWT juga memuat sebuah *library user interface*.dari Widgets (*textbox*, *dropdown menu*, *menu bar, tree control*, *dialog boxes*) dan *panel*, *library* untuk pemanfaatan komunikasi *asynchronous server* melalui HTTP atau RPC, *tool* untuk interoperasi dengan aplikasi web yang lain menggunakan Java Script, JSON dan XML dan yang lainnya.

*Library* yang disediakan GWT memungkinkan komunikasi dengan *server* berlangsung secara tidak sinkron sehingga dapat meningkatkan performa aplikasi.

GWT dalam *portal* berperan sebagai perancang aplikasi untuk menolong *user* secara otomatis melakukan *generate* kode XML yang mendeskripsikan *tasks datamining*. Selain itu sebagai manajer aplikasi yang mengijinkan *user* untuk mendaftar untuk memiliki akses ke *resource* aplikasi dan mengelola semua aksi. Sebuah kode *generator* digunakan untuk secara otomatis menserialisasi objek java ke dan dari JSON dan XML. Aplikasi *client side* menggunakan *resource* komputasi pada *server* menggunakan mekanisme integrasi *server GWT-enabled*. Seperti mengintegrasi dengan *php script* berbasis aksi, atau dengan *GWT-RPC Servlet*.

Penelitian Vasile ini menghasilkan sebuah portal knowledge yang mengintegrasikan fungsionalitas datamining terdistribusi secara lokal atau secara remote sebagai inti dari sistim jaringan enterprise-wide yang mengijinkan end-user untuk berkolaborasi, berbagi informasi secara cepat dengan tool analisis seperti model Data Mining. Terbangunnya aplikasi web yang powerfull yang terlihat sangat mirip dengan aplikasi desktop. Semua fungsionalitas dapat diakses menggunakan web browser standar antara lokasi yang berbeda atau cabang dari sebuah korporasi.

Dalam penelitian Vasile ini terlihat bahwa *web service* dapat dipergunakan untuk mengintegrasikan aplikasi *Data Mining*, dan dapat membuat semua fungsionalitas *Data Mining* dapat diakses menggunakan *browser* standar. *Data Mining*, *Portlet* dan WSRP semua menggunakan platform Java. Apakah sistem dengan platform yang berbeda dapat diintegrasikan?, belum dapat diketahui dalam

penelitian Vasile ini.

Vipul K. Dabhi, dkk (2009) mengintegrasikan prosedur *rechecking/reassessment* ujian mahasiswa pada universitas. Prosedur *rechecking/reassessment* melibatkan tiga proses utama yaitu: a). Mahasiswa membayar dan mengisi formulir untuk *rechecking/reassessment* dengan menyediakan informasinya. b). Mahasiswa memasukkan form yang telah diisi ke fakultas. c). Fakultas akan mengerjakan rechecking/reassessment dan menyediakan status hasilnya.

Prosedur yang ada sebelum diintegrasikan adalah, mahasiswa membayar pada Account Department dan mendapatkan formulir kosong untuk permintaan *rechecking* dan *reassessment*. Account Department merekam nomor, semester dan nama cabang. Account Department memiliki Ms-sql untuk peyimpaanan rekaman. Setelah melakukan prosedur *rechecking/reassessment*. Fakultas secara langsung mengembalikan status mahasiswa. Tidak ada rekaman yang dikelola oleh fakultas.

Untuk menyelesaikan *manual/paper work procedure*, dirancang suatu web service untuk tiap-tiap proses, dan mengintegrasikan *web service* untuk menyedikan penanganan secara otomatis dan online dari prosedur rechecking/reassessment. Aplikasi ini membuat prosedur *rechecking/reassessment* efektif, efisien, akses secara remote dan membantu mengelola rekaman.

Aplikasi menggunakan tiga *web service*: satu *service* berkaitan dengan mahasiswa dan dua yang lain berkaitan dengan fakultas. *Web service* mahasiswa

mengambil request rechecking dan me-retrieve status reassessment. Kedua proses melibatkan verifikasi mahasiswa. Web service mahasiswa dibangun dengan platform JEE dan menggunakan ms-sql database. Dua web service untuk fakultas adalah: a). mendownload request rechecking/reassessment mahasiswa. b). mengupdae status rechecking/reassessment. Yang pertama dibangun dengan platform JEE dan yang kedua dibangun pada platform .NET.

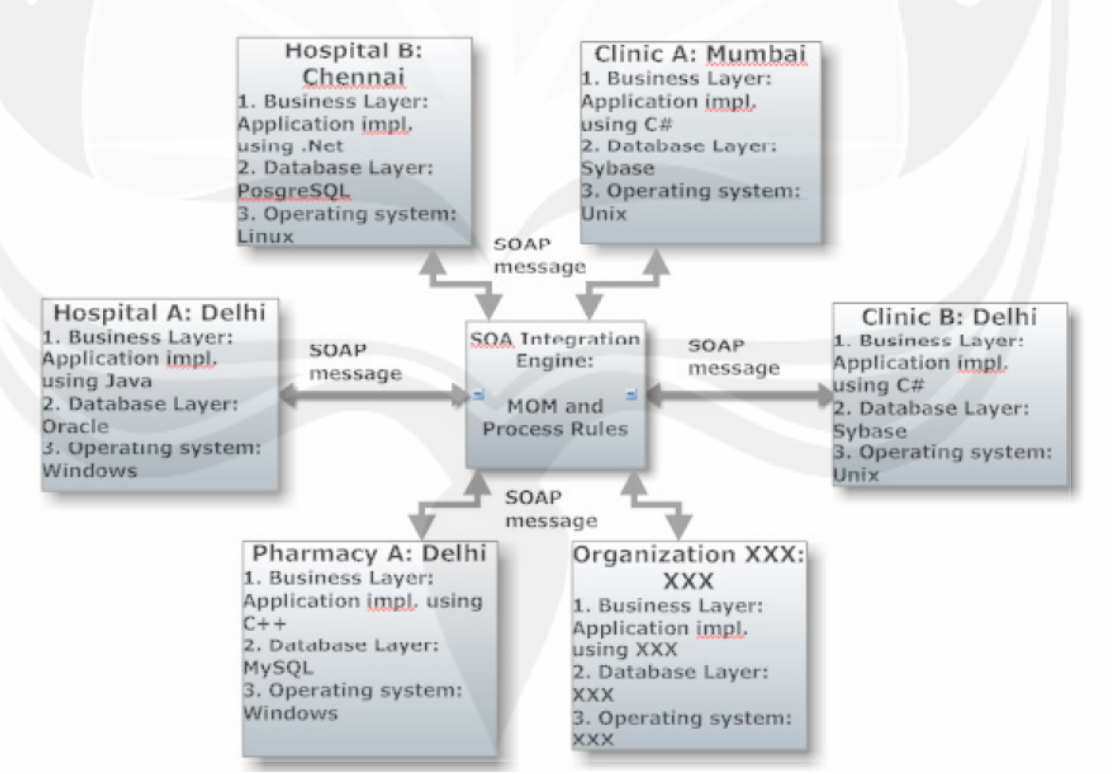
Web service fakultas dibangun menggunakan platform JEE dengan autentifikasi fakultas, retrieve rechecking request dan mengirimkan request ini ke service .NET dan mengupdate request rechecking dengan memodifikasi tanda yang lama. Web service yang dibangun pada .NET memperoleh request dari JEE service dan menampilkan request ini ke fakultas dan menghapus entry dari temporary database pada .NET server.

Aplikasi mengunakan dua database a). Database utama yang mana menyimpan informasi yang berkaitan dengan mahasiswa, fakultas, request rechecking yang disubmit oleh mahasiswa dan status atau hasil dari request rechecking. b). Temporary database digunakan untuk menunjukkan jumlah dari request recehecking/reassessment yang menunggu untuk fakultas. Ketika status atau hasil diupdate oleh fakultas untuk sebuah request rechecking/reassessment tertentu, yang detail request akan dihapus dari temporary database dan diupdate dalam database utama. Dua database diletakkan pada node yang berbeda.

Sama seperti penelitian Vasile, dalam penelitian Vipul dkk tampak bahwa web service dapat dipergunakan untuk integrasi aplikasi. Selain itu, penelitian

Vipul dkk menunjukkan bahwa integrasi dapat dilakukan dengan menggunakan platform web service yang berbeda dengan menggunakan dua jenis basis data yang merupakan produk dari vendor yang sama. Platform basis data yang berlainan vendor tidak tampak pada penelitian Vipul ini.

Integrasi aplikasi juga dilakukan oleh Batra Usha dan Mukharjee Saurabh (2011). Mereka menggunakan web service untuk mengintegrasikan aplikasi-aplikasi yang terdapat pada berbagai rumah sakit. Mereka membuat sebuah *middleware* yang disebut dengan *Message Oriented Middleware* (MOM) yang mengeksekusi pertukaran pesan SOA seperti yang terlihat pada gambar 2 di bawah ini.




Gambar 2. Arsitektur Integrasi Batra Usha (2011)

Teknologi yang digunakan adalah web service dengan menggunakan pesan *SOAP Request* dan *SOAP Response* dalam pertukaran pesan. Langkah pertama, rumah sakit A (Delhi) membuat *request* untuk mendapatkan detail dari pasien pada rumah sakit B (Chennai). Langkah selanjutnya, rumah sakit B (Chennai) mengirim sebuah *response* dengan *transaction id*. *Transaction id* ini dapat digunakan untuk komunikasi lebih lanjut.

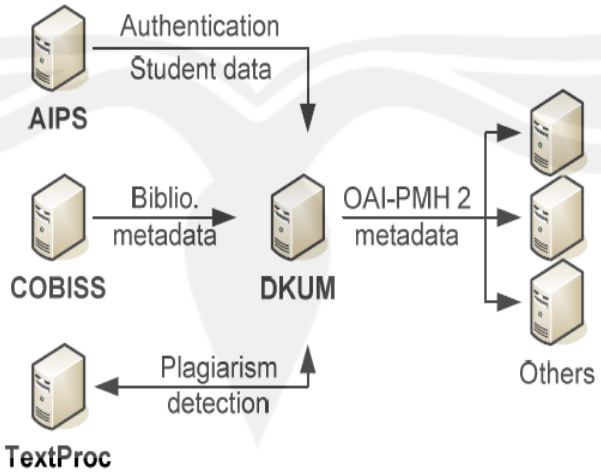
Sedangkan integrasi aplikasi yang berkaitan dengan perpustakaan dilakukan oleh Brezovnik Janez, Ojsteršek Milan (2011) yang mengintegrasikan perpustakaan digital dengan sistem informasi akademik, sistem bibliografi dan sistem pendeteksi plagiarisme pada perpustakaan universitas Maribor, Slovenia. Sistem ini disebut dengan DKUM (Digitalna knjižnica Univerze v Mariboru).

Sistem informasi akademik universitas Maribor menyimpan seluruh data mahasiswa, sistem ini disebut AIPS (Academic Information Sub System). AIPS menangani proses autentifikasi untuk masuk ke DKUM. Komunikasi antara DKUM dan AIPS dilakukan via web service (Brezovnik Janez, Ojsteršek Milan, 2011).

Sistem bibliografi adalah sistem katalog perpustakaan eksternal yang disebut COBISS (Cooperative Online Bibliographic System and Service). Sistem ini merupakan jaringan dari basis data bibiligrafi local, menangani lebih dari 400 perpustakaan lokal di Slovenia, sistem yang sama juga berjalan di Serbia, Bosnia dan Herzegovina, Montenegro, Macedonia dan sedang proses implementasi di beberapa negara lain. Integrasi dengan COBISS dilakukan dengan pertolongan

dari sebuah protocol pertukaran data Z39.50. dengan ini, DKUM mendukung import record COBISS secara penuh, di mana semua metadata dari COBISS disalin ke dalam DKUM, sebagai sebuah *record* baru atau *overwrite record* yang telah ada (Brezovnik Janez, Ojsteršek Milan, 2011). DKUM juga mendukung protocol OAI-PMH V2 (Open Archieve Initiative – Protokol untuk mengambil metadata). Protocol ini berbasis *open source* standar (XML dan HTTP) . DKUM hanya dapat melakukan operasi *read* dan tidak membuat perubahan dalam COBISS.

DKUM memeiliki kemampuan mendeteksi plagiarisme. Hal ini berkat integrasi dengan aplikasi deteksi plagiat yang disebut dengan TextProc. TextProc memiliki kemampuan untuk menjalankan proses ini melalui aplikasi web dan sebagai web service untuk integrasi, jadi DKUM berkomukasi dengan TextProc memalui web service (Brezovnik Janez, Ojsteršek Milan, 2011). Arsitekutur integrasi sistem DKUM dapat dilihat pada gambar 2. di bawah ini.




Gambar 3. Integrasi DKUM (Brezovnik Janez, Ojsteršek Milan, 2011)

## B. LANDASAN TEORI

### 1. Integrasi Aplikasi

Integrasi Aplikasi *Enterprise* merupakan pendekatan strategis untuk membungkus beberapa sistem informasi bersama-sama dan mendukung kemampuan untuk bertukar informasi secara real time (Puustjärvi Juha, Puustjärvi Leena, 2010). Sedangkan menurut Litan D, Velicanu M, Copcea L, dkk 2011, Integrasi Aplikasi *Enterprise* adalah proses mengintegrasikan berbagai aplikasi independen. Dengan menggunakan integrasi aplikasi, berbagi data dan proses dapat dilakukan tanpa melakukan perubahan terdahap aplikasi-aplikasi dan struktur datanya, yang mana hal ini merupakan efektifitas dalam hal pembiayaan (Soomro Tariq Rahim, Awan Abrar Hasnain, 2012).

Terdapat beberapa pendekatan dalam integrasi aplikasi *enterprise* yaitu : *data integration* (integrasi data) yang digunakan untuk sharing informasi sederhana, *method integration* (integrasi metode) merupakan konsep pertukaran berbasis pesan yang disebut juga integrasi aplikasi web. Terakhir adalah *process integration* (integrasi proses) yang disebut juga pendekatan otomasi proses yaitu terdapat penambahan kemampuan seperti *process modeling* dan simulasi aliran kerja. (Samuel S. Justin, Sasipraba T, 2010).

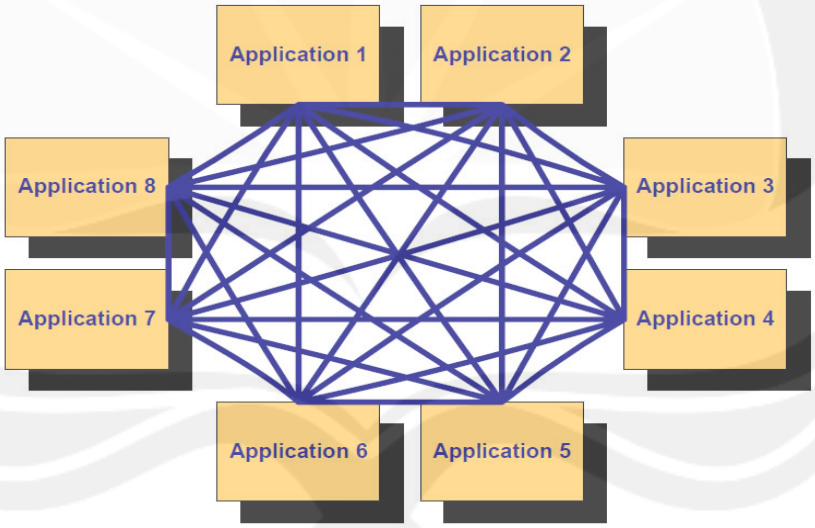
Solusi integrasi aplikasi *enterprise* menyediakan fungsionalitas melalui arsitektur integrasi menggunakan teknologi seperti *database-oriented middleware, message oriented technologies, transaction based technologies, distributed object technologies,* dan *interface oriented technologies* (Kamal Muhammad, 2010).

Sedangkan menurut Rehan Mohammad dan Akyuz Goknur Arzu, 2010 teknologi yang digunakan antara lain *Java Message Service* (JMS), *Remote Method Invocation* (RMI), *Component Object Model* (COM), *Distributed Object Model* (DCOM), *Common Object Request Broker Architecture* (CORBA) dan *Web Services* (Rehan Mohammad, Akyuz Goknur Arzu, 2010), (Puustjärvi Juha, Puustjärvi Leena, 2010). *Web service* merupakan solusi untuk mengintegrasikan sumber-sumber informasi yang tersebar, mandiri dan heterogen (Akaichi Jalel, Limam Hela, 2011).

Strategi integrasi aplikasi *enterprise* antara lain *Database oriented, process oriented, service oriented* dan *portal oriented integration* (Puustjärvi Juha, Puustjärvi Leena, 2010). Integrasi data atau strategi *Database oriented* adalah mengkonsolidasi data dari sumber-sumber data yang berbeda ke dalam satu basis data (Helena Vranesic dan Christoph Rosenkranz, 2009). Integrasi data adalah penyediaan antar muka yang tunggal yang seragam untuk mengakses sumber-sumber data yang berbeda (Craven Robert, Lobo Jorge, dkk, 2009), (Mehta Hemant, Kanungo Priyesh, dkk, 2010), (Lu Jing, Peng Dunlu, dkk, 2010). Terdapat 4 model dalam membangun integrasi aplikasi *enterprise* yaitu *Point-to-Point , Hub-and-Spoke, Messaging BUS* dan *Enterprise Service BUS with BPM* (Jujian Zhang, 2009).

Point-to-point merupakan model integrasi aplikasi yang paling tua, model ini digunakan untuk menyelesaikan masalah beberapa sistem *enterprise* yang saling terpisah. Yang mana pertukaran data menjadi tidak handal sehingga

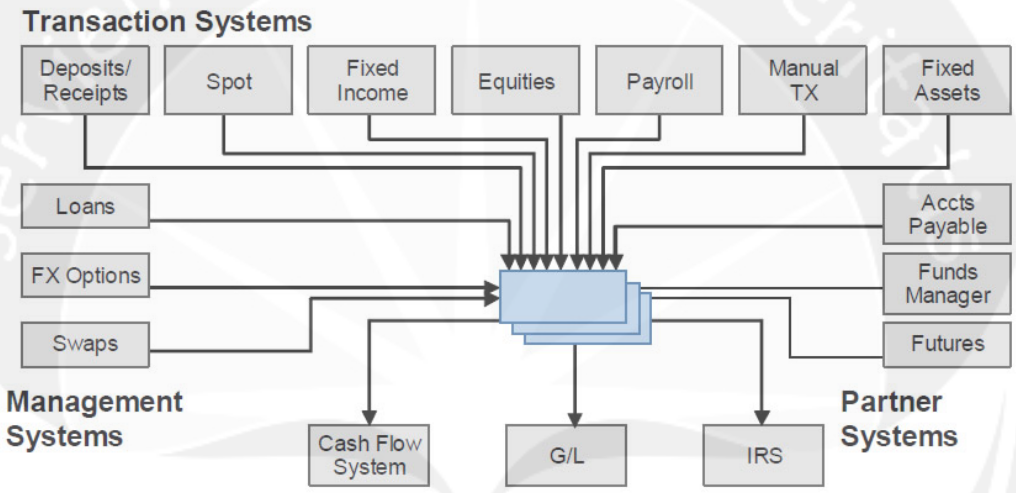
dibutuhkan secara manual untuk melakukannya. Idenya adalah membangun *interface object* dari dua sistem (*point*) yang diperlukan untuk melakukan pertukaran data. Data yang sesuai diambil oleh *interface object* dari sistem yang menjadi sumber data, untuk menyediakan sistem aplikasi tujuan. Karakteristik dari sistem ini adalah tujuan *interface object* yang jelas, penyederhanaan transmisi format data, dan *performance* transmisi data yang tinggi. Tetapi memiliki *interface* yang kompleks dan penyambungan yang berlebihan, sehingga pengelolaannya menjadi sulit. Skema integrasi *point to point* dapat dilihat pada gambar berikut ini.




Gambar 4. Model *point to point* (Risimic dejan, 2007)

Setelah model point-to-point, model peralatan distribusi (star) diperkenalkan, model ini diperuntukkan untuk mengatasi *multi-interface* dan penyambungan yang berlebihan yang terdapat pada model *point-to-point.* Model ini menggunakan *distributed control system* (DCS) dan *Adapter,* yang mana

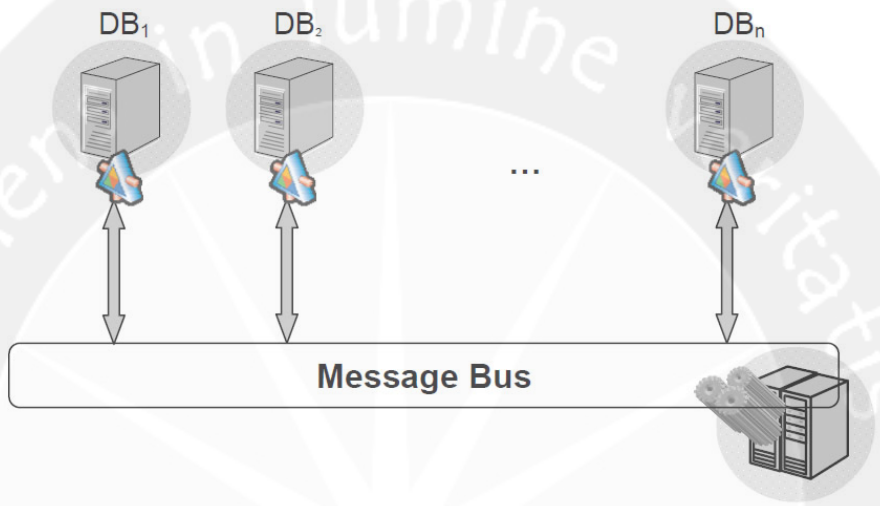
memiliki konsep beberapa sistem akses ke perangkat sentral melalui *adapter.* Sehingga hanya ada satu *link* dalam arsitektur tersebut ke luar sistem. Tetapi redundansi data yang berbasis pada sinkronisasi data dibandingkan atas aplikasi berbasis bisnis merupakan sambungan yang lebih rapat dan memerlukan jaringan yang lebih tinggi. Model ini disebut sebagai *model hub and spoke (broker)* . Skema model *hub and spoke* dapat dilihat pada gambar berikut ini.




Gambar 5. Model *hub and spoke* (Swithinbank Peter dkk, 2007)

Model *message bus* bertujuan untuk menyelesaikan masalah penyambungan berlebih, konektifitas dan skalabilitas. Ini menggunakan mekanisme pesan tak sinkron, yang mana memiliki banyak sistem yang terhubung ke *messaging bus* melalui *adapter* dan *converter.* Semua data dan informasi ditransfer ke dalam form dari *news. Message initiator* bertanggung jawab untuk melakukan *posting news* ke dalam *message bus,* dan *news consumer* (pengguna) berlangganan *news* tertentu dari *message bus* untuk mendapatkan sistem penyambungan yang lebih sedikit. Karena *message bus* menggunakan teknologi

*publish/subscribe* untuk merespon pesan, maka kinerja sistem menurun di tingkat tertentu dan tidak dapat dicapai benar-benar secara simultan dan menjadi sulit untuk mencapai transaksi global antar berbagai sistem. Skema model *message bus* dapat dilihat pada gambar berikut ini.




Gambar 6. Model *message bus* (Kiselyova Nadezhda, dkk, 2008)

**2. Web Service**

Akhir-akhir ini, *web service* untuk integrasi aplikasi *enterprise* telah menjadi topic penelitan yang luas (Su Xiaoyong dkk, 2009). *World Wide Web Consortium* (W3C) mendefinisikan web service sebagai perangkat lunak sistem yang dirancang untuk mendukung interaksi antara mesin dengan mesin melalui jaringan (Lucky, 2008), (Bassil Youssef, 2012**)**. Teknologi web service menyediakan kemampuan untuk mengintegrasikan jenis sistem dan aplikasi yang berbeda tanpa melihat platform, sistem operasi, bahasa pemrograman ataupun lokasi (Tripathi Sandesh, Abbas S Q, Beg Rizwan, 2011). Web service menggunakan standar industri, web service mengenkapsulasi aplikasi dan

mempublikasinya sebagai *service* (Reddy Ch Ram Mohan, dkk, 2011).

Paradigma baru dalam web service adalah *Service-Oriented Architecture* (SOA) (Mardukhi Farhad, NematBaksh Naser, Zamanifar Kamran, 2011). Teknologi *web service* dan SOA menawarkan kesempatan yang lebih dalam integrasi aplikasi *enterprise* dengan penambahan keuntungan dari pengurangan biaya, kemudahan perawatan, fleksibilitas yang lebih besar dan skalabilitas yang dapat ditingkatkan (Mahmood Zaigham, 2007).

Pertukaran pesan dalam web service menggunakan format *Simple Object Access Protocol* (SOAP) yang fleksibel (Kuehnhausen Martin, 2010). *Web service* mengijinkan aplikasi yang berbeda dari sumber yang berbeda untuk saling berkomunikasi tanpa memakan waktu untuk melakukan kustomisasi kode, karena semua komunikasi dalam bentuk XML (Thirumaran M dkk, 2011). Penghubung web service didefinisikan, diuraikan dan ditemukan dengan XML, dukungan interaksi pesan berbasis *eXtensible Markup Language* (XML) dengan aplikasi lain menggunakan protocol berbasis internet seperti SOAP (Khapre Shailesh, Chandramohan D, 2011).

Web service berbasis SOAP kadang-kadang disebut juga Web Service berbasis XML. Semua pertukaran pesan antar web service menggunakan format XML. Artinya paremeter input dan data output pada web web service dalam format XML. Perintah *client* untuk menggunakan web service tertentu membutuhkan suatu uraian *description* yaitu *Web Service Description Language* (WSDL) untuk mengetahui method yang tersedia, parameter apa yang diperlukan

oleh method tersebut dan data jenis apa yang dikembalikan (Asberg Mikael, Stromback Lena, 2010). *XML Web Service* adalah fungsi logika yang dapat diakses dengan menggunakan protokol standar yaitu HTTP. *XML Web Service* juga merupakan sebuah fungsi yang dapat mengintegrasikan berbagai sistem dari bermacam bahasa pemrograman serta *basis data* yang berbeda (Wicaksono Soetam Rizky, 2008).

**3. XML**

XML merupakan bahasa yang digunakan untuk membuat dokumen yang mana komputer dan manusia dapat membaca dengan mudah dokumen tersebut (Firat Mehmet, Kuzu Abdullah, 2011). XML merupakan bahasa *mark-up* yang ditujukan untuk menjabarkan informasi dengan struktur yang dapat dibuat (Kadir Abdul, 2009). XML merupakan dasar terbentuknya web service. web service dapat berkomunikasi dengan aplikasi-aplikasi yang memanggilnya dengan menggunakan XML. XML bersifat *platform independent* sehingga informasi di dalamnya dapat dibaca oleh aplikasi dan *platform* yang berbeda selama aplikasi tersebut mampu menerjemahkan tag-tag XML (Lucky, 2008).

Dengan demikian XML merupakan kunci untuk suatu platform independen dan pertukaran data menggunakan bahasa yang umum (Gupta Siddharth, Thakur Narina, 2010). XML dirancang sebagai bahasa untuk mark-up atau catatan tambahan dari suatu dokumen. XML memisahkan struktur data dari data itu sendiri (Banu Ayesha dkk, 2011). Contoh dokumen XML yang dipergunakan untuk autentifikasi *user* dan *password* dapat dilihat pada gambar 7

berikut ini (Shanmughaneethi V, dkk, 2011).

```
<?xml version="1.0"?>
<Login xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance
 xsi: noNamespaceSchemaLocation = "authenticate.xsd">
<user>
        <uname>computer</username>
        <passwd>centre</passwd>
</user>
<user>
        <uname>asia</username>
        <passwd>india</passwd>
</user>
<user>
        <uname>tamil</username>
        <passwd>chenai</passwd>
</user>
```

Gambar 7. Contoh dokumen XML (Shanmughaneethi V, dkk, 2011)

**4. SOAP**

*Simple Object Access Protocol* (SOAP) merupakan format standar dokumen berbentuk XML yang digunakan untuk melakukan proses *request* dan *response* antara *web service* dengan aplikasi yang memanggilnya (lucky, 2008). Dokumen SOAP yang dipergunakan untuk melakukan *request* disebut SOAP *request*, sedangkan dokumen SOAP yang diperoleh dari *web service* disebut SOAP *response*. SOAP merupakan teknologi yang menyediakan web *service* yang memungkinkan untuk berhubungan dengan program yang lain meskipun program tersebut ditulis menggunakan bahasa pemrograman yang berbeda dan dijalankan pada komputer yang berbeda (Gashti Mehdi Zekriyapanah, 2012).

Struktur SOAP terdiri atas sebuah SOAP *envelope* yang didalamnya terdapat SOAP *header* dan SOAP *body*. SOAP *header* berisi informasi-informasi dokumen SOAP sedangkan SOAP *body* berisi informasi yang akan dipertukarkan dalam *web service*.

**5. WSDL**

Web Service Description Language (WSDL) adalah sebuah dokumen dalam format XML yang isisnya menjelaskan informasi detail sebuah *web service.* dalam WSDL dijelaskan parameter-paremeter yang diperlukan untuk memanggil sebuah *method,* dan tipe data yang dikembalikan oleh *method* yang dipanggil tersebut (Lucky, 2008). WSDL merupakan sintaks bahasa XML sebagai alat penghubung web service (Cesare Pautasso, Olaf Zimmermann, Frank Leymann, 2008).