BAB 3
Web Service

Seperti telah dijelaskan sebelumnya, SOA terdiri atas sekumpulan layanan. Menurut Luthria et al, (2009), jika layanan mencerminkan fungsi bisnis di dalam model komputasi berbasis layanan, maka SOA menyediakan kerangka kerja untuk infrastruktur yang memudahkan interaksi dan komunikasi antar layanan. SOA dapat dipandang sebagai arsitektur maupun model pemrograman, lebih merupakan cara berpikir mengenai pengembangan perangkat lunak daripada teknik pengembangan perangkat lunak. Hampir serupa dengan Luthria et al, (2009), Mueller et al, (2010) juga menyatakan bahwa SOA merupakan paradigma arsitektur terdistribusi, *multi layer*, yang membungkus bagian dari sistem informasi sebagai layanan.

SOA juga merupakan paradigma perangkat lunak yang berlaku secara dinamis untuk mengintegrasikan layanan secara longgar ke dalam satu proses bisnis yang kohesif menggunakan kerangka kerja komponen perangkat lunak secara standard. Sistem berbasis SOA dapat mengintegrasikan layanan *legacy* maupun layanan baru, yang dibuat perusahaan dan dipasang di dalam atau di luar penyedia layanan (Lin et al, 2009). Menurut Kart et al (2008) SOA memungkinkan penggunaan ulang dari komponen perangkat lunak,

menyediakan protokol bebas, dan memudahkan integrasi aplikasi. SOA menguatkan prinsip-prinsip arsitektur perangkat lunak dalam hal perancangan modular, abstraksi dan enkapsulasi. Dengan standard terbuka seperti XML dan SOAP, SOA menyediakan interoperabilitas antar layanan yang bekerja pada platform berbeda dan antar aplikasi yang diimplementasikan dalam bahasa pemrograman yang berbeda.

3.1 Arsitektur Web Service

Dalam hal ini WS merupakan sebuah teknologi yang dapat digunakan untuk mengimplementasikan layanan . (Salter-Jennings, 2008). Menurut Hwang et al, (2008), WS secara fakta telah menjadi standard untuk melakukan ekspose fungsi dari aplikasi bisnis, WS akan menjadi blok bangunan untuk pengembangan aplikasi generasi mendatang menggunakan SOA. Walaupun arsitektur SOA saat ini mendukung *registry*, *discovery* dan konsumsi WS, namun cara efektif mengintegrasikan beberapa WS ke dalam sebuah komposit tetap merupakan sebuah tantangan dan menarik perhatian dari industri dan akademis.

WS telah banyak dipakai untuk membangun aplikasi berbasis SOA. Malah dengan penggunaan WS ini, potensi-potensi SOA baru dapat dimunculkan. Hal inilah yang menyebabkannya WS identik dengan SOA. Selain itu, perluasan dari WS yang disebut dengan WS-*, memberikan pengaruh terhadap perkembangan SOA. Akibatnya, SOA semakin identik dengan WS. Atas dasar ini Erl (2005) menyebutnya sebagai *Contemporary* SOA. *Contemporary* SOA merupakan SOA yang menggunakan WS dan XML

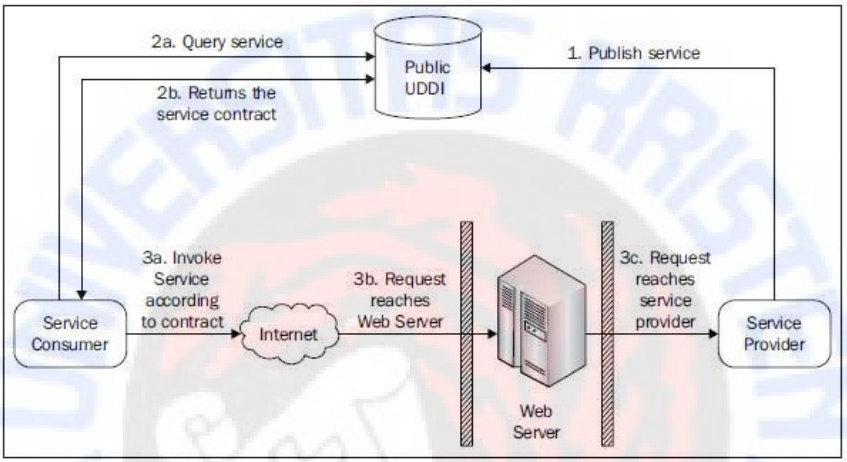
dalam implementasinya. Untuk selanjutnya, yang disebut sebagai SOA adalah *Contemporary* SOA.

Definisi yang diajukan oleh W3C tentang WS adalah: WS merupakan aplikasi perangkat lunak yang diidentifikasi melalui URI yang antarmuka dan *binding* nya mampu diidentifikasi, dideskripsikan dan ditemukan melalui XML dan mendukung interaksi langsung dengan aplikasi perangkat lunak lain menggunakan pesan berbasis XML melalui protokol berbasis internet.

Perkembangan WS yang begitu cepat menyebabkan lahirnya ekstensi-ekstensi WS yang memperluas fungsi WS itu sendiri. Untuk membedakan WS dasar dengan ekstensi-ekstensinya, maka keduanya dibedakan menjadi WS generasi pertama dan WS generasi kedua. WS generasi pertama adalah pondasi dari teknologi WS ini sendiri. Implementasi dari teknologi ini, yang berupa aplikasi WS, saling berinteraksi satu sama lain dengan menggunakan dokumen berformat XML dan protokol pengiriman pesan SOAP (*Simple Object Access Protocol*) melalui HTTP. Format XML, SOAP, dan HTTP ini juga merupakan standar terbuka yang dapat diadopsi (Erl, 2004, Kumar, 2010).

Perkembangan teknologi WS berlanjut pada munculnya ekstensi-ekstensi dari WS, yang biasa disebut generasi kedua WS atau WS-* (Erl, 2004, Kumar, 2010). Ekstensi ini merupakan pengembangan dari teknologi WS dasar yang muncul disebabkan kebutuhan-kebutuhan yang ada di dalam perusahaan. Adapun beberapa ekstensi yang jamak digunakan adalah WS–Coordination,

WS–Transaction, *Business Process Execution Language for Web services* (BPEL4WS atau WS-BPEL), *WS-ReliableMessaging*, *WS–Addressing*, *WS–Policy*, *WS–PolicyAssertions*, *WS– PolicyAttachments*, *WS –Attachments*, dan *SOAP with Attachments* (SwA).



Gambar 3.1 Arsitektur Web Service (Juric et al, 2007)

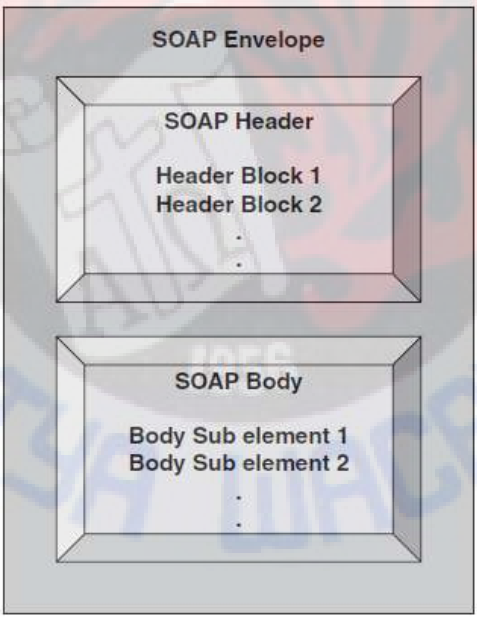
Komponen WS terdiri dari 3 yaitu SOAP, WSDL dan UDDI (Gambar 3.8). Penjelasan dari ketiga komponen tersebut akan diuraikan dibawah ini.

1. SOAP

Dalam komunikasi antara WS diperlukan suatu standard format pesan antara peminta layanan dengan penyedia layanan. SOAP adalah format pesan yang digunakan untuk komunikasi tersebut. SOAP mendefinisikan mekanisme pembungkusan yang mengatur pertukaran pesan antara WS. Pesan SOAP adalah dokumen XML

yang mengandung tiga elemen yaitu envelope, header, body. Struktur umum SOAP dapat dilihat pada Gambar 3.9.

Envelope adalah elemen utama dari pesan SOAP. Elemen ini mengandung elemen *header* yang opsional dan elemen *body* yang harus ada. Elemen *header* adalah mekanisme umum untuk mendefinisikan fitur tambahan pada SOAP. Elemen *body* berisi deskripsi pesan aktual ditujukan untuk penerima akhir dan akan diproses. Komunikasi pesan SOAP antar WS ini ilakukan melalui protokol HTTP.



Gambar 3.2 Elemen SOAP (Erl et al, 2008)

2. WSDL

WSDL adalah spesifikasi dari W3C yang menyediakan bahasa untuk mendeskripsikan definisi dari WS (Erl et al, 2008). Dalam definisi tersebut termasuk juga deskripsi layanan dan fungsifungsi yang disediakan oleh WS.

Gambar 3.10 menunjukkan ada 5 elemen WSDL yaitu types, *message*, portType, binding, dan *service*/port yang menjelaskan layanan (Erl et al, 2008).

- ***types*** Elemen ini mendefinisikan type data yang terdapat dalam pesan yang dipertukarkan sebagai bagian dari layanan. Type data berupa simple, complex, derviced, atau array. Type (baik definisi skema maupun reference) yang diacu dalam elemen dokumen pesan WSDL didefinisikan dalam elemen type dokumen WSDL.
- ***message*** Elemen ini mendefinisikan pesan dari layanan yang dipertukarkan. Dokumen WSDL mempunyai elemen pesan yang tiap pesannya dipertukarkan dan elemen pesan berisi type data yang diasosiasikan dengan pesan.

Gambar 3.3 Lima elemen WSDL (Erl et al, 2008)

- **portType** Elemen ini menetapkan secara abstrak, operasi dan pesan yang merupakan bagian dari layanan. Dokumen WSDL mempunyai satu atau lebih definisi portType.
- **binding** Elemen ini melakukan bind type port abstrak dan pesan dan operasinya, untuk protokol transport dan untuk format pesan.
- ***service* and port** Elemen-elemen ini bersama-sama mendefinisikan nama layanan aktual dan menyediakan address tunggal untuk menetapkan end point individual dari layanan. Port dapat mempunyai satu alamat. Kelompok elemen layanan berkaitan dengan port dan melalui atribut nama, menyediakan nama logikal untuk layanan.

3. *UDDI Registry*

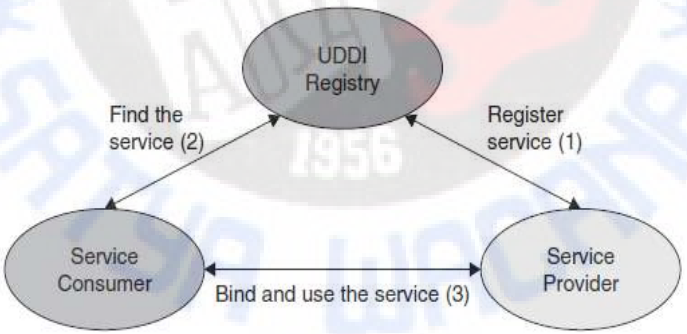
UDDI adalah spesifikasi yang telah diakui secara luas untuk pendaftaran suatu WS. UDDI mendefinisikan metadata dan protokol yang digunakan untuk mengetahui dan mengubah informasi dari suatu WS. Langkah pertama dalam menemukan WS adalah dengan meminta alamat tempat penyimpanan dari WS yang akan dipakai yang biasa disebut dengan direktori (Gambar 3.11).

Setelah menemukan direktori, peminta WS dapat mengirimkan permintaan lagi untuk mendapatkan informasi detail tentang WS (misalnya penyedia WS dan dimana diletakkan). Selanjutnya perangkat lunak menggunakan informasi yang didapat untuk secara dinamik mengakses WS yang diinginkan.

Tempat penyimpanan UDDI dapat dibagi dalam tiga cara, yaitu :

1. *Public UDDI*, dianggap sebagai resource bagi WS berbasis internet. Salah satu contohnya adalah UDDI Business Registry (UBR) yang dimiliki oleh grup yang terdiri dari IBM, Microsoft, SAP.
2. *Intra Enterprise UDDI*, merupakan tempat penyimpanan yang private yang menyediakan layanan bagi kalangan internal suatu perusahaan.
3. *Inter Enterprise UDDI*, merupakan repository yang dapat diakses oleh perusahaan yang merupakan partner dalam suatu bisnis.

Seperti yang telah dijelaskan terdahulu, penemuan layanan memainkan peranan yang penting dalam SOA. Banyak cara yang dapat dipakai untuk mekanisme ini, tapi dalam teknologi WS, UDDI menyediakan standard yang baik dan fleksibel untuk penemuan layanan web.



Gambar 3.4 UDDI (Erl, 2005)

UDDI mengorganisasikan pendaftaran dalam enam tipe data (Erl, 2004):

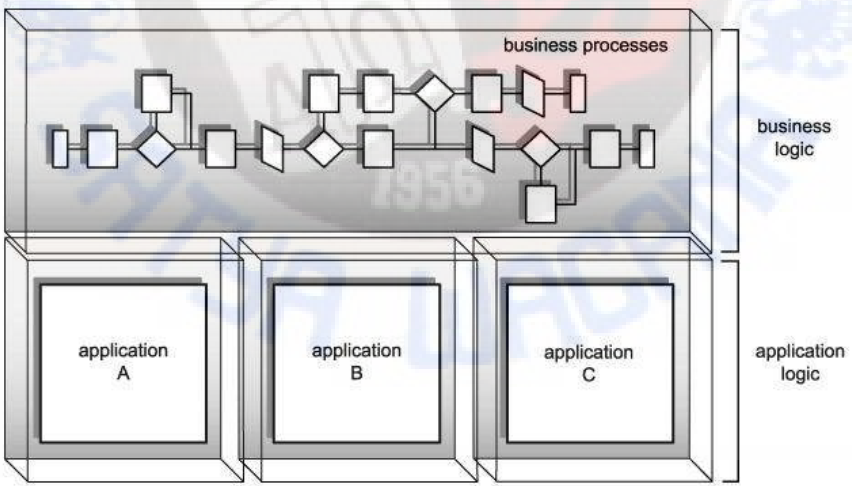
1. *Business Entity*, menyimpan informasi profil mengenai bisnis yang disimpan, termasuk nama, deskripsi, dan *unique identifier*.
2. Layanan bisnis, merepresentasikan layanan aktual yang ditawarkan oleh bisnis terdaftar, disimpan dalam elemen businessEntity
3. *Specification Pointers*, menyimpan halaman dari layanan bisnis ke informasi implementasi, disebut juga *Binding Components*
4. *Service types*, menyediakan informasi dari definisi internal
5. *Business relationships*, direpresentasi dengan publisherAssertion, menyimpan hubungan antara entitas bisnis dengan yang lainnya
6. *Subscriptions*, direpresentasi dengan elemen subscription, memungkinkan *subscriber* mendapatkan notifikasi saat profil *business entity* diperbaharui.

3.2 Integrasi Web Service

Erl (2005) mengajukan metode integrasi yang dinamakan SOAD. Dalam metode ini gap antara lojik bisnis dan aplikasi diisi oleh Lapisan Antarmuka Layanan yang berisi tiga lapisan yaitu Lapisan Layanan Aplikasi, Proses Bisnis dan Orkestrasi. Siklus hidup SOAD diterapkan pada pengembangan Lapisan Antarmuka Layanan. Siklus hidup berisi enam fase yaitu *service-oriented analysis*, *service-oriented design*,

service development, *service testing*, *service deployment*, dan *service administration*. Fase-fase ini mirip dengan proyek pengembangan berorientasi obyek kecuali pada pengenalan pertimbangan unik pada setiap fase dari konstruksi dan pengiriman layanan. Misalnya pada fase analisis berorientasi layanan berisi tiga langkah 1) mendefiniskan skope analisis, 2) mengidentifikasi sistem otomatisasi dan 3) memodelkan kandidat layanan.

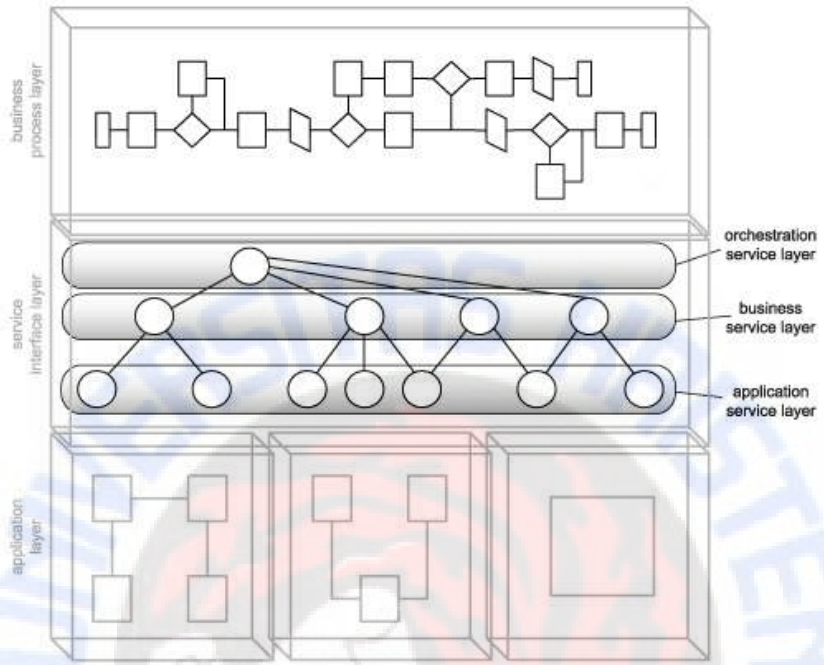
Seperti yang telah diuraikan pada latar belakang masalah, perangkat lunak yang tidak menggunakan SOA secara umum dapat dibagi menjadi dua lapisan utama, yaitu Lapisan Aplikasi tempat aplikasi dijalankan dan Lapisan Proses Bisnis yang mendeskripsikan cara proses bisnis dalam perusahaan berjalan. Proses bisnis organisasi akan didefinisikan dalam aplikasi bersamaan dengan kode program yang bersifat teknis. Hal tersebut dapat dilihat pada Gambar 4.1.



Gambar 3.5 Lapisan lojik bisnis dan lojik aplikasi dalam SOAD (Erl, 2005)

Dalam implementasi SOA, konsep berorientasi layanan diimplementasikan dalam sebuah lapisan di antara Lapisan Proses Bisnis dan Lapisan Aplikasi yang keduanya merupakan bagian dari lojik perusahaan. Lapisan tersebut dinamakan Lapisan Antarmuka Layanan, dan dapat dilihat pada Gambar 4.2. Fungsi dari lapisan ini adalah membungkus lojik yang ada di lojik aplikasi, sekaligus proses bisnis yang ada di lojik bisnis. Dengan pendekatan ini, aplikasi bisa lebih dimodularisasi dan menggunakan berbagai macam teknologi.

Lapisan Antarmuka Layanan ini juga terbagi atas 3 lapisan abstraksi, yaitu Lapisan Layanan Aplikasi, Lapisan Layanan Bisnis, dan Lapisan Layanan Orkestrasi. Ilustrasi dari ketiganya dapat dilihat pada Gambar 4.2.



Gambar 3.6 Lapisan Antarmuka Layanan pada SOA (Erl, 2005; Shirazi et al, 2009; Lee et al, 2010)

Lapisan Layanan Aplikasi

Dari Gambar 3.6, Lapisan Layanan Aplikasi. menyediakan sekumpulan layanan yang spesifik untuk mengenkapsulasi teknologi tertentu yang terdapat di dalam lojik aplikasi. Layanan yang disediakan dalam lapisan ini akan melakukan abstraksi terhadap semua lojik yang tidak terkait dengan proses bisnis, namun dibutuhkan untuk menjalankan fungsi-fungsi yang ada di proses bisnis tersebut. Misalkan sebuah proses bisnis mengandung aktivitas notifikasi yang mengharuskan pengiriman

email kepada pihak yang dinotifikasi, maka Lapisan Layanan Aplikasi menyediakan layanan untuk mengirimkan email.

Lapisan Layanan Bisnis

Pada Gambar 3.6, Lapisan Layanan Bisnis merepresentasikan lojik bisnis dari aplikasi. Lapisan ini bisa diibaratkan sebagai *controller* dari Lapisan Layanan Aplikasi. Pada Lapisan Layanan Bisnis inilah fungsi-fungsi bisnis, yang berupa aktivitas-aktivitas yang dilakukan untuk menjalankan proses bisnis disediakan. Aktivitas-aktivitas tersebut akan menjalankan fungsi-fungsi yang ada di Lapisan Layanan Aplikasi jika dibutuhkan. Lapisan ini sendiri menyediakan abstraksi kepada pengguna layanannya mengenai fungsi-fungsi bisnis yang ada, sehingga tidak perlu melihat bagaimana lojik aplikasi dijalankan di sana.

Ada dua macam layanan yang mungkin disediakan oleh Lapisan Layanan Bisnis, yaitu:

1. *Task-centric*, yaitu sebuah layanan yang mengenkapsulasi lojik tertentu dalam proses bisnis yang melibatkan satu atau dua entitas bisnis. Layanan jenis ini memiliki potensi penggunaan ulang yang terbatas, namun proses analisis untuk menghasilkannya akan lebih mudah
2. *Entity-centric*, yaitu layanan yang mengenkapsulasi lojik yang sifatnya spesifik terhadap entitas bisnis tertentu. Layanan jenis ini

sangat baik dalam penggunaan ulang, namun proses analisis yang dibutuhkan untuk menghasilkannya lebih sulit.

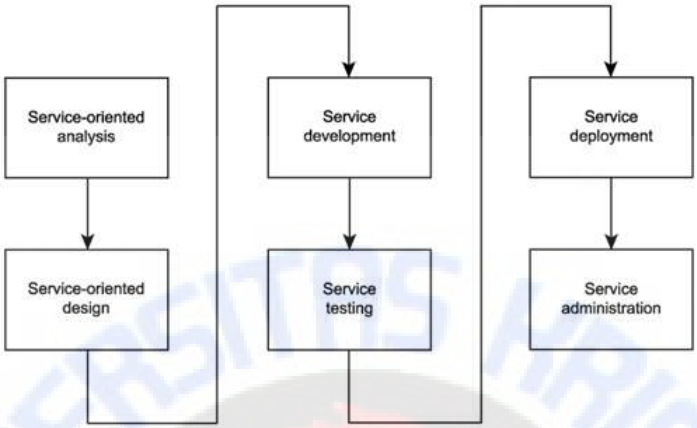
Lapisan Layanan Orkestrasi

Lapisan Layanan Orkestrasi pada Gambar 6.2 merupakan sebuah lapisan yang menyediakan abstraksi dengan level tertinggi dari aplikasi. Pada lapisan ini semua proses bisnis yang ada di dalam sistem didefinisikan dan dijalankan dengan menggunakan fungsi-fungsi yang terdapat pada Lapisan Layanan Bisnis.

Walaupun siklus hidup SOAD menyediakan panduan umum yang bagus untuk pengembang yang membangun Lapisan Antarmuka Layanan, namun tidak ada teknik pemodelan dan *reverse engineering* dari perusahaan yang sudah ada. Selain itu SOAD juga tidak menyediakann dokumentasi UML.

Siklus Hidup SOAD

Siklus hidup SOAD yang diperkenalkan Erl (2005) mengikuti fase-fase pada Gambar 3.7.



Gambar 3.7 Fase dalam SOAD (Erl, 2005)

Skema fase pengembangan aplikasi dengan menggunakan SOAD dapat dilihat dalam Gambar 3.7. Fase pengembangan dalam SOAD terbagi menjadi 6 fase yaitu *service-oriented analysis*, *service oriented design*, *service development*, service *testing*, *service deployment*, dan *service administration*. Walaupun terbagi menjadi 6 fase, inti dari SOAD dapat dilihat dari 2 fase pertama, yaitu *service-oriented analysis* dan *service-oriented design* karena dari 2 fase pertama inilah konsep dan prinsip mengenai berorientasi layanan mulai dimasukkan dalam analisis dan perancangan solusi yang akan dibuat.

Service-Oriented Analysis

Service-oriented analysis adalah fase awal dalam pengembangan sebuah aplikasi SOA, dimana akan ditentukan lingkup dari aplikasi SOA yang akan dibuat. Setelah lingkup aplikasi

ditentukan, selanjutnya dilakukan identifikasi dan analisis layanan yang akan ada di dalam aplikasi.

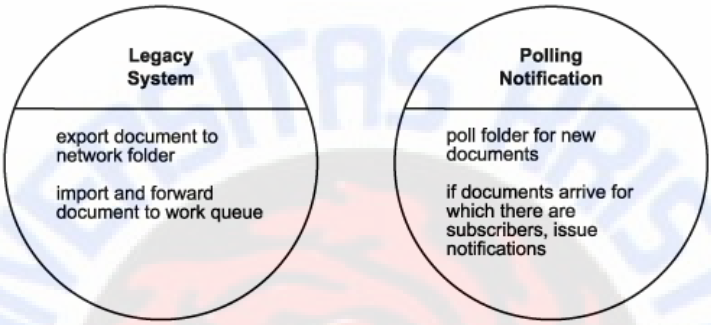
Tujuan yang ingin dicapai dalam tahapan *service-oriented analysis* adalah sebagai berikut (Erl, 2005):

1. Mendefinisikan kandidat-kandidat layanan yang akan ada
2. Mengelompokkan kandidat layanan yang telah didefinisikan dalam sesuai dengan konteks lojik masing-masing
3. Mendefinisikan batasan antara layanan sehingga tidak terjadi *overlap* antar layanan.
4. Mengidentifikasi lojik yang terkandung di dalam layanan yang berpotensi untuk digunakan kembali
5. Memastikan bahwa lojik dalam tiap layanan sudah sesuai dengan tujuan dan fungsionalitas dari layanan yang bersangkutan
6. Mendefinisikan model awal komposisi layanan

Service-oriented analysis terbagi menjadi 3 langkah, yaitu (Erl, 2005) :

1. Mendefinisikan kebutuhan bisnis, dengan model Diagram *Activity*
2. Mengidentifikasi sistem yang telah ada, yaitu melihat dan melakukan pendataan sistem lain pada lingkungan operasional aplikasi yang mempunyai kemungkinan untuk berinteraksi atau memberikan pengaruh kepada aplikasi nantinya.
3. Memodelkan kandidat-kandidat layanan yang disediakan oleh aplikasi. Contoh dari kandidat layanan dan operasi layanan yang dihasilkan dapat dilihat dalam Gambar 3.8. Kandidat layanan yang

dimaksud di sini adalah kandidat layanan dalam bentuk layanan bisnis dan layanan aplikasi, serta lojik orkestrasi dalam bentuk layanan proses.



Gambar 3.8. Contoh Kandidat Layanan dan Operasi pada Layanan (Erl, 2005)

Service Oriented Design

Setelah membuat kandidat dari layanan pada tahap analisis, maka tahap berikutnya adalah membuat perancangan konkrit dari kandidat layanan yang telah ada dan mengimplementasikannya dalam sebuah komposisi yang membentuk suatu proses bisnis. Dalam SOAD, perancangan dan implementasi dari layanan menggunakan teknologi *web service* yang berbasis *XML* (*eXtensible Markup Language*), *XSD* (*XML Schema Definition*), SOAP (*Simple Object Access Protocol*), dan WSDL, dan WS-* *extension*. (Erl, 2005)

Tujuan yang ingin dicapai dalam kegiatan *service oriented design* ini adalah sebagai berikut (Erl, 2005):

1. Mendefinisikan secara fisik antarmuka layanan dari kandidat layanan yang telah didapatkan dalam *service oriented analysis*
2. Menentukan karakteristik SOA yang akan diimplementasikan
3. Menentukan standar yang diperlukan oleh aplikasi SOA yang dibuat

Tujuan yang telah disebutkan di atas akan dirinci lagi dalam poin-poin sebagai berikut (Erl, 2005):

1. Menentukan bagian utama dari sistem
2. Menentukan standar apa saja yang akan dipakai dalam sistem yang akan dibuat
3. Membuat batasan yang jelas dari sistem yang akan dibuat
4. Mendefinisikan antarmuka layanan
5. Mengidentifikasi kemungkinan komposisi layanan yang mungkin muncul
6. Menerapkan prinsip berorientasi layanan
7. Mengeksplorasi dukungan untuk sistem SOA lainnya

Langkah-langkah yang ada pada *service oriented design* ada 3 langkah utama, yaitu (Erl, 2005):

1. Mengkomposisi SOA, yaitu mendefinisikan teknologi yang akan digunakan dalam SOA. Dalam SOAD, teknologi SOA yang dipergunakan adalah WS.

2. Merancang layanan, yang mencakup penentuan hasil akhir layanan-layanan berupa layanan bisnis *entity-centric*, layanan bisnis *task-centric*, dan layanan aplikasi berdasarkan kandidat layanan yang telah didapatkan dalam tahap analisis.

3. Merancang proses bisnis. Proses bisnis yang akan didesain pada langkah ini adalah proses bisnis untuk lapisan orkestrasi yang akan menentukan lojik aliran kerja dari SOA.

Dalam mendesain layanan baik *entity-centric*, *task-centric*, dan layanan aplikasi, langkah yang dilakukan adalah sebagai berikut (Erl, 2005) :

1. Menentukan layanan dari tahap analisis yang akan didesain
2. Menentukan skema pesan yang digunakan.
3. Mendefinisikan antarmuka dari layanan. Antarmuka disini berupa *port* yang dapat mengirimkan dan menerima pesan. *Port* akan didefinisikan dengan menggunakan WSDL, lalu untuk tiap *port* akan didefinisikan pesan yang akan diterimanya dengan menggunakan *schema* yang telah disepakati.
4. Mengaplikasikan prinsip berorientasi layanan, yang berupa *reusablity*, *autonomy*, *statelessness*, dan *discoverability*
5. Me-*refine* ulang antarmuka layanan yang telah didefinisikan
6. Mengembangkan lebih lanjut operasi yang ada dalam suatu layanan sehingga mencakup semua hal yang mungkin.
7. Mengidentifikasi batasan teknis yang ada, baik dari segi keamanan, performansi, dan realibilitas

8. Mengidentifikasi layanan lain yang diperlukan, apabila layanan yang didesain bersifat kompleks dan merupakan hasil komposisi dari banyak layanan.