BAB 2

LANDASAN TEORI

2.1 Sistem Informasi

O'Brien (2002, p8) berpendapat bahwa sistem adalah kumpulan komponen yang saling terkait dan bekerja sama untuk pencapaian tujuan dengan menerima masukan dan menghasilkan keluaran di dalam sebuah proses transformasi atau perubahan yang terorganisasi.

Menurut Turban, Rainer, dan Potter (2006, p52), data adalah penjelasan dasar atas segala sesuatu, peristiwa, aktivitas, dan transaksi yang dicatat, diklasifikasi serta disimpan, tetapi tidak diatur untuk mengungkapkan makna tertentu.

Berdasarkan pendapat Turban, Rainer, dan Potter (2006, p52), informasi adalah data yang telah diatur sehingga memiliki makna dan nilai bagi penerimanya.

Sistem informasi adalah pengaturan orang, data, proses, dan teknologi informasi yang berinteraksi untuk mengumpulkan, memproses, menyimpan, dan menyediakan keluaran informasi yang diperlukan untuk mendukung sebuah organisasi (Whitten, Bentley, dan Dittman, 2004, p10).

2.2 Basis Data

2.2.1 Basis Data

Connolly dan Begg (2005, p15) menjelaskan bahwa basis data adalah kumpulan data yang terhubung secara logis yang digunakan bersama-sama dan

deskripsi dari data tersebut yang dirancang untuk memenuhi kebutuhan informasi sebuah organisasi.

Menurut Connolly dan Begg (2005, pp72-84, p356), ada beberapa istilah penting dalam basis data, antara lain:

1. Relasi: sebuah tabel dengan kolom dan baris.
2. Atribut: kolom yang diberi nama pada sebuah relasi.
3. *Relationship*: asosiasi antar tabel.
4. *Primary key*: *candidate key* yang terpilih untuk mengidentifikasi *tuple* secara unik dalam sebuah relasi.
5. *Multiplicity*: jumlah kejadian yang mungkin dari sebuah tipe entitas yang berhubungan dengan kejadian tunggal dari tipe entitas lain yang berhubungan melalui *relationship* tertentu.

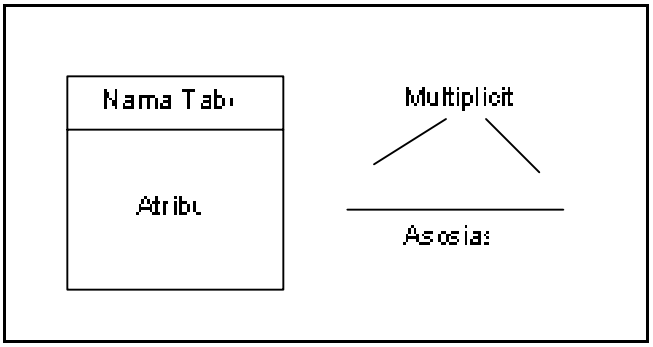
Lebih lanjut, Connolly dan Begg (2006, p362) membahas tentang batasan *multiplicity* yang diringkas dalam Tabel 2.1.

Tabel 2.1 Batasan *Multiplicty*

(Sumber: Connolly dan Begg, 2006, p362)

Batasan *Multiplicity*	Arti
0..1	Kejadian entitas nol atau satu.
1..1 (atau hanya 1)	Kejadian entitas tepat satu.
0..* (atau hanya *)	Kejadian entitas nol atau lebih.
1..*	Kejadian entitas satu atau lebih.
5..10	Kejadian entitas minimum 5 sampai maksimum 10
0, 3, 6-8	Kejadian entitas nol atau tiga atau enam atau tujuh atau delapan.

Connolly dan Begg (2006, p15) menerangkan bahwa relasi-relasi yang berhubungan dihubungkan menggunakan asosiasi untuk membentuk sebuah diagram yang dinamakan *Entity Relationship Diagram* (ERD). Notasi yang digunakan pada ERD dapat dilihat pada Gambar 2.1.



Gambar 2.1 Notasi *Entity Relationship Diagram*

(Sumber: Connolly dan Begg, 2006)

2.2.2 *Database Management System* (DBMS)

Connolly dan Begg (2005, pp16-17) menerangkan bahwa *Database Management System* (DBMS) adalah sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke basis data.

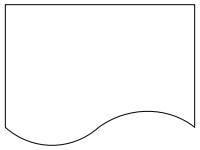
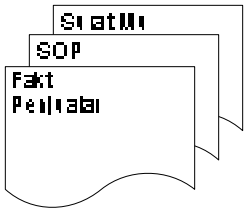
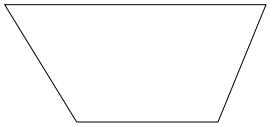
2.3 Diagram Aliran Dokumen (DAD)

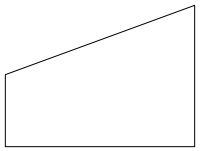
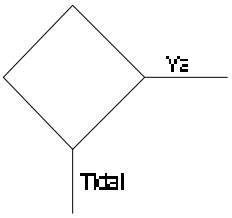
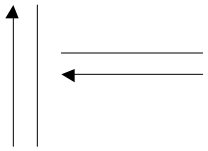
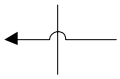
Menurut Mulyadi (2001, pp58-63), Diagram Aliran Dokumen (DAD) adalah suatu model yang menggambarkan aliran dokumen dan proses untuk

mengolah dokumen dalam suatu proses. Komponen-komponen dari DAD dapat dilihat pada Tabel 2.2.

Tabel 2.2 Notasi Diagram Aliran Dokumen

(Sumber: Mulyadi, 2001, pp58-63)

Simbol	Keterangan
	Dokumen Simbol ini digunakan untuk menggambarkan semua jenis dokumen yang merupakan formulir untuk merekam data terjadinya suatu transaksi.
	Berbagai dokumen Simbol ini digunakan untuk menggambarkan berbagai jenis dokumen yang digabungkan bersama di dalam satu paket. Nama dokumen dituliskan di dalam masing-masing simbol dan nomor lembar dicantumkan di sudut kanan atas simbol dokumen yang bersangkutan.
	Kegiatan manual Simbol yang digunakan untuk menggambarkan kegiatan manual.
	*On-line computer process* Simbol ini menggambarkan pengolahan data dengan komputer secara *on-line*. Nama program ditulis di dalam simbol.

Simbol	Keterangan
	*Keying* (*typing, verifying*) Simbol ini menggambarkan pemasukan data ke dalam komputer melalui *on-line terminal.*
	Keputusan Simbol ini menggambarkan keputusan yang harus dibuat dalam proses pengolahan data. Keputusan yang dibuat ditulis dalam simbol.
	Garis Alir Simbol ini menggambarkan arah proses pengolahan data. Anak panah tidak digambarkan jika dokumen mengarah ke bawah dan ke kanan. Jika arus dokumen mengarah ke atas atau ke kiri, anak panah perlu dicantumkan.
	Persimpangan Garis Alir Jika dua garis alir bersimpangan, untuk menunjukkan arah masing-masing garis, salah satu garis dibuat sedikit melengkung tepat pada persimpangan kedua garis tersebut.
	Pertemuan Garis Alir Simbol ini digunakan jika dua garis alir bertemu dan salah satu garis mengikuti garis lainnya.

Simbol	Keterangan
	Mulai / Berakhir (*terminal*) Simbol ini untuk menggambarkan awal dan akhir suatu sistem.
	Masuk ke sistem Karena kegiatan di luar sistem tidak perlu digambarkan dalam bagan alir, maka diperlukan simbol untuk menggambarkan masuk ke sistem yang digambarkan dalam bagan alir.
	Keluar ke sistem lain Karena kegiatan di luar sistem tidak perlu digambarkan dalam bagan alir, maka diperlukan simbol untuk menggambarkan ke luar sistem lain.

2.4 Proses Pengembangan Sistem

Proses pengembangan sistem adalah satu set aktivitas, metode, praktik terbaik, baranng siap dikirim, dan peralatan terotomasi yang digunakan pada stakeholder untuk mengembangakan dan secara berkesinambungan memperbaiki sistem informasi dan perangkat lunak (Whitten, Bentley, dan Dittman, 2004, p78).

Menurut Lenz, dan Moeller (2004, p25), sebelum memilih sebuah model pengembangan sistem, penting untuk mencari model yang cocok dengan proyek dan lingkungan di mana model akan dilokasikan.

2.5 *Service Oriented Architecture* (SOA)

Service Oriented Architecture (SOA) merupakan sebuah representasi model baru untuk membangun aplikasi yang terdistribusi (Hasan, 2004, p1). Menurut Brown (2008), SOA adalah sebuah gaya arsitektural yang memodularisasi sistem informasi menjadi *services*. Berdasarkan pendapat Bieberstein et.al. (2008), SOA adalah sebuah framework yang mengintegrasikan proses bisnis dan mendukung infrastruktur IT yang aman, berkomponen terstandarisasi (*services*) yang dapat digunakan kembali dan disertakan dalam prioritas bisnis yang berubah.

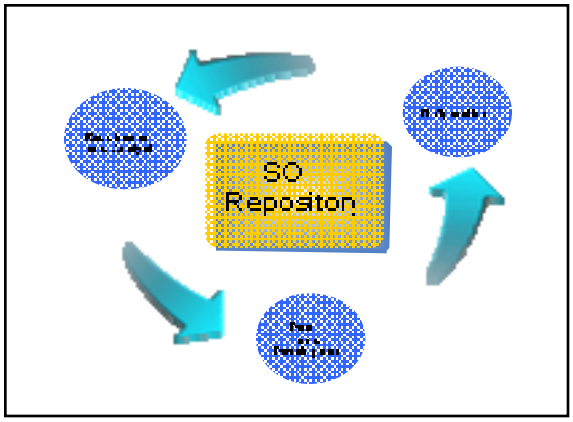
Menurut Thomas Erl (2005), terdapat beberapa aspek kunci pada prinsip SOA, yaitu :

1. *Loose coupling*, yaitu bahwa *services* tersebut mempertahankan sebuah hubungan yang meminimalisasi ketergantungan dan mereka hanya perlu menjaga kesadaran antar satu sama lain.
2. *Service contract*, *services* melekat dan taat pada sebuah kesepakatan komunikasi, yang didefinisikan secara kolektif oleh satu atau lebih deskripsi service dan dokumen yang berhubungan.
3. *Autonomy*, bahwa *services* mempunyai kendali berdasarkan logika yang dienkapsulasi.

4. *Abstraction*, di luar apa yang dideskripsikan pada *service contract*, *services* menyembunyikan logika dari dunia luar.
5. *Reusability*, logika terbagi menjadi *services* dengan tujuan untuk digunakan kembali.
6. *Composability*, kumpulan dari *services* dapat dikoordinasikan dan dihimpun untuk membentuk *services* yang berbeda.
7. *Statelessness*, *services* meminimalisasi sifat berpegang teguh pada informasi tertentu untuk sebuah aktivitas.
8. *Discoverability*, *services* dirancang dengan sifat yang deskriptif sehingga mereka dapat ditemukan dan ditentukan dengan menggunakan mekanisme-mekanisme penemuan yang ada.

2.5.1 SOA *Lifecycle*

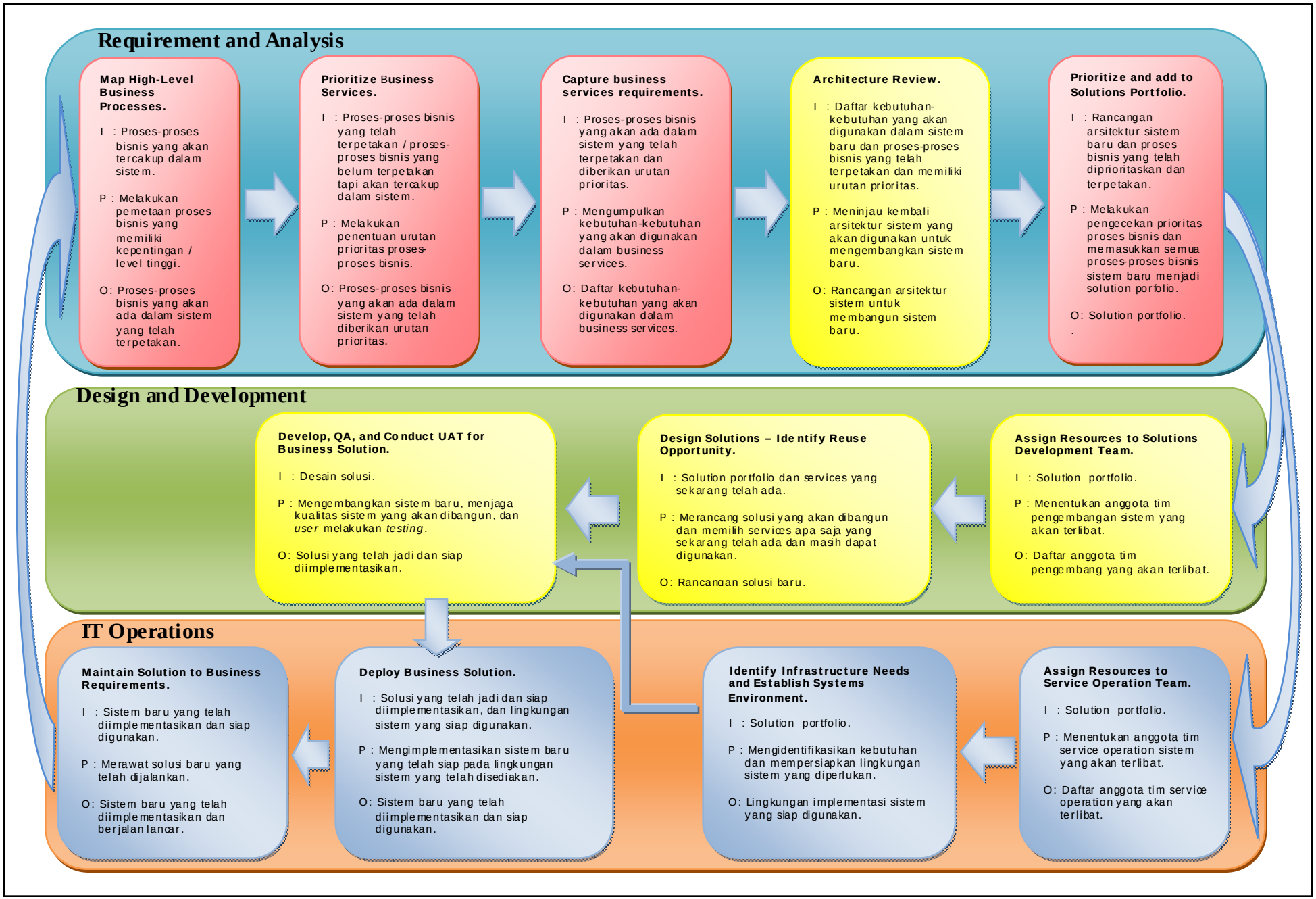
Menurut Durvasula (2006, p9), siklus hidup *service* memungkinkan penempatan kemampuan *service* melalui tiga tahap, yaitu : *requirements and analysis*, *design and development*, dan *IT operations*.



Gambar 2.2 Service Lifecycle

(Sumber: Duvasula, 2006, p10)

Gambar untuk keseluruhan siklus besar pada SOA dapat dilihat pada Gambar 2.3.



Gambar 2.3 Keseluruhan Siklus Besar Tahapan Metode Pengembangan Sistem Informasi dengan SOA

Tiga proses utama yang terjadi dalam satu siklus SOA, dapat terbagi-bagi lagi menjadi beberapa subproses dan keseluruhan akan berjumlah 12 subproses, yaitu :

1. *Requirement and Analysis* :
   a. *Map High-Level Business Processes:*

      Ini merupakan tahap awal dari tahap *requirements and analysis*. Pada tahap ini dilakukan pemetaan terhadap proses-proses bisnis yang akan tercakup dalam sistem. Proses-proses bisnis yang akan tercakup dalam sistem dianalisis dan dipetakan. Pada akhir tahap ini akan menghasilkan proses-proses bisnis yang akan tercakup dalam sistem yang telah terpetakan.

   b. *Prioritize Business Services :*

      Pemetaan proses-proses bisnis dalam sistem yang telah dikerjakan pada tahap *Map High-Level Business Processes*, kemudian dilanjutkan dengan melakukan penentuan urutan prioritas proses-proses bisnis sehingga dihasilkan proses-proses bisnis yang akan tercakup dalam sistem yang telah diberikan urutan prioritas.

   c. *Capture business services requirements :*

      Untuk masing-masing proses bisnis yang telah memiliki urutan prioritas, kemudian dilakukan pengumpulan kebutuhan-kebutuhan yang akan digunakan dalam *business services,* sehingga diperoleh daftar kebutuhan-kebutuhan yang akan digunakan dalam *business services*.

   d. *Architecture Review :*

Pada tahap ini dilakukan peninjauan kembali arsitektur yang akan digunakan untuk mengembangkan sistem baru dengan menggunakan daftar proses-proses bisnis yang akan tercakup dalam sistem dan telah terpetakan beserta dengan daftar kebutuhan pada masing-masing proses bisnis.

Architecture Views (Gambar 2.4) adalah gambaran dari keseluruhan aritektur yang penuh arti yang berguna bagi satu atau lebih *stakeholders* dalam sistem. Sebuah arsitektur menggambarkan satu atau lebih model arsitektur yang bergabung membentuk sebuah deskripsi yang koheren atas arsitektur sistem yang akan dibangun.

Gambar 2.4 *Architecture Views*

Sudut pandang arsitektur yang ditinjau kembali meliputi 4 aspek, yaitu :

i. *Business Architecture Views* :

Business Architecture Views menunjuk pada hal-hal yang harus diperhatikan oleh *User*, *Planner* dan *Business Manager*, dan fokus pada aspek fungsionalitas sistem dari segi perspektif *User*, hal-hal yang termasuk di dalamnya adalah performa, fungsionalitas sistem, dan kegunaan sistem. Sebuah implementasi SOA bisa dikatakan berhasil bila SOA diarahkan pada *business architecture*. Kemampuan untuk digunakan kembali dari sebuah proses bisnis akan memberikan nilai *Return On Investments* (ROI) yang lebih tinggi dibandingkan kemampuan penggunaan kembali dari sebuah infrastruktur atau komponen data yang potensial.

Aspek-aspek dalam *Business Architecture Views*, antara lain :

- *People View* :

  Fokus pada aspek sumber daya manusia yang terlibat dalam sistem.

- *Business Process View* :

  Berhubungan dengan proses-proses bisnis yang tercakup dalam sistem.

- *Business Function View* :

  Berhubungan dengan fungsi yang diperlukan untuk mendukung masing-masing proses bisnis dalam sistem.

- *Business Information View* :

Berhubungan dengan informasi yang dibutuhkan untuk mendukung dalam alur proses bisnis.

- *Usability View* :

  Memperhatikan aspek kegunaan dari sistem dan lingkungan sistem yang akan terbentuk.

- *Business Performance View* :

  Memperhatikan aspek performa sistem dan performa lingkungan sistem.

ii. *Data Architecture Views* :

Arsitektur data berhubungan dengan model logikal dan fisikal data yang akan digunakan. Termasuk dalam hal ini adalah tipe dan panjang data yang akan digunakan untuk merepresentasikan data yang akan disimpan. Dengan arsitektur data yang baik, maka data yang disimpan juga akan memiliki kualitas yang baik.

iii. *Application Architecture Views* :

Berhubungan dengan bagaimana cara memodelkan setiap bagian yang akan terdapat dalam aplikasi. *Application Architecture Views* mengarahkan terbentuknya komunikasi yang baik antar komponen dalam aplikasi namun tidak memiliki tingkat ketergantungan yang tinggi.

iv. *Technology Architecture Views* :

Menunjuk pada kepedulian *acquirers, operators, communications engineers, administrators,* dan *manager*. Beberapa aspek dalam *Technology Architecture Views*, antara lain :

- *Communications Engineering View* :

  Menunjuk pada hal-hal yang diperhatikan oleh seorang *communications engineer*. Aspek ini akan menguji beberapa cara yang bervariasi untuk menyusun fasilitas komunikasi guna menyederhanakan perencanaan dan perancangan jaringan bisnis.

- *Acquirer's View* :

  Menyediakan sebuah panduan yang cocok untuk pendayagunaan komponen. *Acquirer's View* berhubungan dengan biaya dan standard-standard yang harus diikuti untuk mengarah pada efektifitas biaya.

Pada akhir tahap ini dihasilkan rancangan arsitektur untuk membangun sistem baru.

e. *Prioritize and add to Solutions Portfolio :*

Tahap ini merupakan tahap akhir dari *requirements and analysis*. Pada tahap ini dimasukkan data-data yang telah dianalisa dan rancangan arsitektur untuk sistem baru. Semuanya akan digabungkan menjadi *solutions portfolio*.

2. *Design and Development* :

a. *Assign Resources to Solutions Development Team :*

Awal dari tahap *design and development* dimulai dengan proses *Assign Resources to Solutions Development Team*. Tim pengembang merupakan salah satu aspek penting untuk pengembangan sistem baru. Setelah memperoleh semua data mengenai kebutuhan untuk pengembangan sistem baru, tahap berikutnya adalah melakukan penentuan anggota tim

pengembangan sistem yang akan terlibat sehingga diperoleh daftar anggota tim pengembang yang akan terlibat.

b. *Design Solutions – Identify Reuse Opportunity :*

Tahap ini merupakan tahap perancangan solusi yang akan dibangun. Hal ini dilakukan dengan mengacu pada *solutions portfolio* dan *services* apa saja yang telah ada sekarang yang masih dapat digunakan.

c. *Develop, QA, and Conduct UAT for Business Solution :*

Setelah memperoleh rancangan solusi baru yang akan dikembangkan, maka berikutnya adalah tahap melakukan pengembangan sistem baru, menjaga kualitas sistem yang akan dibangun, dan memberikan kesempatan kepada *user* untuk melakukan *testing*. Pada tahap pengembangan sistem, digunakan pula pendekatan *Agile Software Development* untuk mendukung perancangan arsitektur aplikasi. *Develop, QA, and Conduct UAT for Business Solution* merupakan tahap akhir dari tahap *design and development.* Pada akhir tahap ini dihasilkan solusi baru yang telah jadi dan siap diimplementasikan.

3. *IT Operations* :

a. *Assign Resources to Service Operation Team :*

Tahap ini merupakan awal dari tahap IT *operations*. Tahap ini adalah tahap melakukan penentuan anggota tim *service operation* sistem yang akan terlibat. Pada akhir tahap ini diperoleh daftar anggota tim *service operation* yang akan terlibat.

b. *Identify Infrastructure Needs and Establish Systems Environment :*

Pada tahap ini dilakukan identifikasi kebutuhan lingkungan sistem dan mempersiapkan segala hal yang diperlukan sehingga diperoleh lingkungan implementasi sistem yang siap digunakan.

c. *Deploy Business Solution :*

Setelah tahap pengembangan sistem baru selesai dan persiapan untuk implementasi sistem telah terpenuhi, maka langkah berikutnya adalah melakukan implementasi sistem baru. Setelah tahap ini diselesaikan, diperoleh sistem baru yang telah diimplementasikan dan siap digunakan.

d. *Maintain Solution to Business Requirements :*

Tahap ini merupakan akhir dari satu siklus besar SOA, yaitu akhir dari tahap IT *operations*. Namun apabila setelah tahap ini berakhir dan terjadi perubahan proses bisnis pada sistem, maka dimungkinkan untuk dilakukan pengulangan dalam langkah-langkah tahap pengembangan sistem baru. Tahap ini merupakan tahap perawatan solusi baru yang telah dijalankan dan memungkinkan untuk dilakukan persiapan untuk pengembangan sistem berikutnya.

2.5.2 *Services*

Jika sebuah SOA hendak diimplementasikan dengan efektif, maka diperlukan pemahaman yang jelas mengenai pengertian *service*. Sebuah *service* adalah sebuah fungsi yang didefinisikan dengan baik, berisi mengenai dirinya sendiri, dan tidak bergantung pada konteks atau situasi *services* lainnya (Anonim 1). Sedangkan *services* adalah segala sesuatu yang terhubung menggunakan *web*

services. Sebuah *service* mengandung beberapa tipe sistem komputer sehingga mendukung koneksi yang terjadi (Anonim 2).

2.6 *Agile Software Development*

Menurut Martin dan Micah (2006), *agile software development* merupakan sebuah model pengembangan sistem yang memungkinkan tim pengembang sistem untuk mengembangkan sistem dan responsif terhadap perubahan dengan cepat, di dalamnya terdapat beberapa nilai utama yang dikandung, yaitu:

1. *Individuals and interactions over processes and tools*
2. *Working software over comprehensive documentation*
3. *Customer collaboration over contract negotiation*
4. *Responding to change over following a plan*

Terdapat beberapa prinsip utama dalam *agile software development*, yaitu:

1. Prioritas utama kami adalah untuk memuaskan pelanggan melalui penyelesaian perangkat lunak yang berkualitas yang dilakukan dengan cepat dan diberikan secara berkelanjutan.
2. Menyambut dengan terbuka terjadinya perubahan kebutuhan sistem, meskipun pada akhir pengembangan. Proses *agile* memanfaatkan perubahan sebagai keuntungan yang bersaing bagi pelanggan.
3. Menyerahkan piranti lunak yang telah beroperasi kepada pelanggan secara berjangka, dari beberapa minggu sampai dengan beberapa bulan, dengan sebuah kecenderungan pada jangka waktu yang lebih pendek.

4. Pelaku bisnis dan pengembang sistem harus bekerja secara bersama sehari-hari di dalam proyek.
5. Melaksanakan pengerjaan proyek di sekitar individu yang bermotivasi. Menyediakan bagi mereka sebuah lingkungan dan memenuhi kebutuhan mereka, serta percaya bahwa mereka dapat menyelesaikan pekerjaan tersebut.
6. Metode yang paling efisien dan efektif dalam menyampaikan informasi kepada dan dengan sebuah tim adalah percakapan tatap muka.
7. Piranti lunak yang dapat beroperasi adalah ukuran utama dari perkembangan yang telah dilaksanakan.
8. Proses-proses agile mengajukan sebuah pengembangan yang dapat bertahan. Para sponsor, pengembang sistem *(developers)*, dan para pengguna harus mampu mempertahankan sebuah langkah yang terus-menerus dan tidak terbatas.
9. Perhatian yang berkelanjutan pada keunggulan teknikal dan rancangan yang bagus menambah tingkat kelincahan sistem.
10. Kesederhanaan keahlian untuk memaksimalisasi jumlah kerja bukanlah hal yang penting.
11. Arsitektur, persyaratan sistem dan rancangan yang terbaik, muncul dari tim yang dapat mengurus dirinya sendiri.
12. Pada sebuah selang waktu yang teratur, tim akan berpikir bagaimana untuk menjadi lebih efektif, kemudian menyesuaikan irama perilakunya sesuai dengan hal tersebut.

    (Robert, Micah, 2006).

2.6.1 *Single-Responsibility Principle* (SRP)

Menurut Robert dan Micah (2006), *Single-Responsibility Principle* memiliki definisi :

> "A class should have only one reason to change."

Jika sebuah class memiliki lebih dari satu fungsi (*responsibility*), maka fungsi-fungsi tersebut akan menjadi bergantung. Perubahan pada sebuah fungsi akan melemahkan atau menghentikan kemampuan *class* tersebut untuk memenuhi fungsi yang lain. Hal ini mengarah pada desain yang rapuh yang akan hancur pada cara yang tidak diharapkan ketika perubahan dilakukan (Robert, Micah, 2006).

2.6.2 *Open/Closed Principle* (OCP)

Menurut Robert dan Micah (2006), *Open/Closed Principle* memiliki definisi :

> "Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification."

Jika OCP dilakukan dengan baik, perubahan tambahan dilakukan dengan penambahan kode, tanpa merubah kode yang sebelumnya telah ada yang telah berjalan (Robert, Micah, 2006).

2.7 *Unified Modeling Language* (UML)

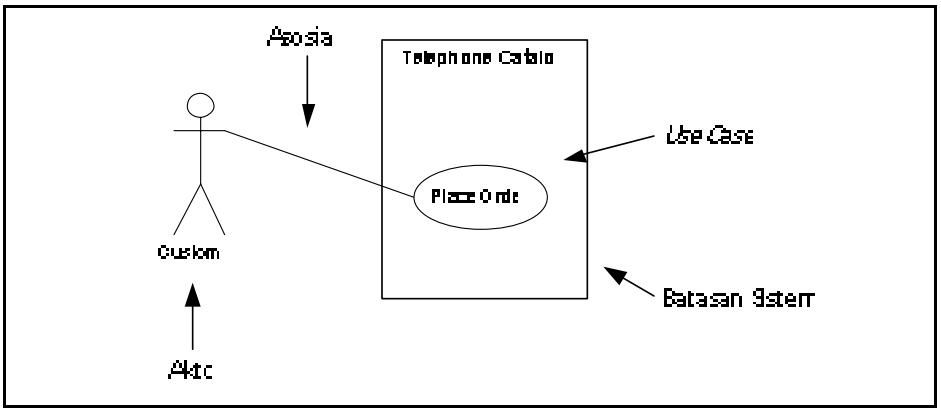
Menurut Rumbaugh, Jacobson, dan Booch (1999, p3), *Unified Modeling Language* (UML) merupakan bahasa pemodelan visual yang bersifat *general-purpose* yang digunakan untuk menspesifikasikan, memvisualisasikan,

membangun, dan mendokumentasikan artifak-artifak dari sebuah sistem perangkat lunak. UML menangkap keputusan-keputusan serta pemahaman mengenai sistem yang akan dibangun. UML ditujukan untuk penggunaan pada semua metode pengembangan, tahapan daur hidup, *domain* aplikasi, dan media.

2.7.1 *Use Case Diagram*

Menurut Whitten, Bentley, dan Dittman (2004, p257) Use Case Diagram adalah diagram yang secara grafis menggambarkan siapa yang akan menggunakan sistem dan dengan cara apa pengguna mengharapkan untuk berinteraksi dengan sistem.

Menurut Rumbaugh, Jacobson, dan Booch (1999, p63), *Use Case Diagram* menangkap perilaku dari sistem ataupun subsistem yang dilihat dari sudut pandang pengguna. Sebuah *use case* menggambarkan interaksi antara aktor dengan sistem. Simbol-simbol yang digunakan dalam *Use Case Diagram* yang dapat dilihat pada Gambar 2.4.



Gambar 2.5 Notasi *Use Case Diagram*

(Sumber: Rumbaugh, Jacobson, dan Booch, 1999, p64)

Hal-hal yang perlu diperhatikan dalam mendokumentasikan *use case* (Schneider, 2001, pp27-47) adalah sebagai berikut:

1. Aktor.

   Aktor adalah segala sesuatu yang berinteraksi dengan sistem. Setiap aktor memiliki peran tertentu.

2. *Precondition*.

   Precondition mengindikasikan apa yang terjadi sebelum *use case* dan menyatakan kondisi sistem pada saat *use case* dimulai.

3. *Post condition*.

   Post condition mengindikasikan apa yang terjadi sesudah *use case* dan menyatakan kondisi sistem pada akhir *use case*. *Post condition* harus selalu bernilai benar dengan alternatif *use case* apapun.

4. *Flow of events*.

   Flow of events merupakan serangkaian pernyataan deklaratif yang menyatakan langkah-langkah *use case* dari sudut pandang aktor.

5. *Basic path*.

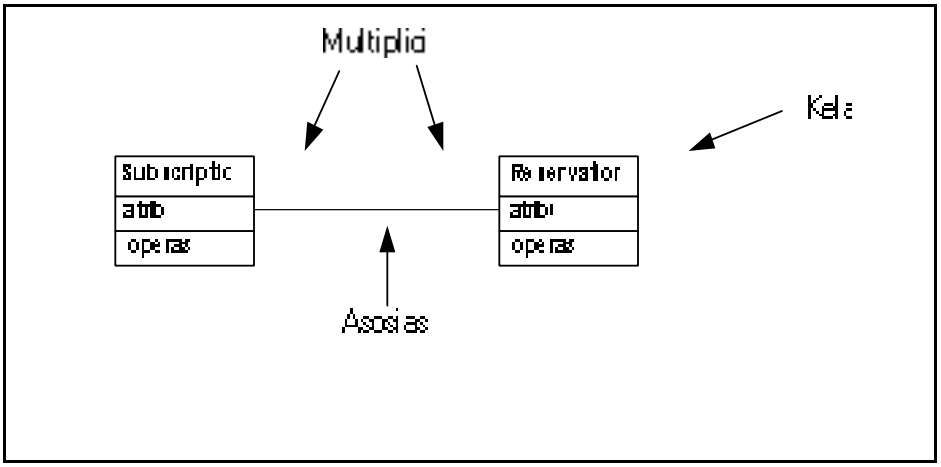
   Basic path digunakan untuk kondisi dimana segala hal berjalan dengan benar. Harus ada satu *basic path* untuk setiap skenario.

6. *Alternative path*.

   Sebuah *alternative path* merupakan sesuatu yang memungkinkan urutan kejadian yang berbeda dengan yang terjadi pada *basic path*.

2.7.2 *Class Diagram*

Rumbaugh, Jacobson, dan Booch (1999, p190) menjelaskan bahwa *Class Diagram* adalah diagram yang menunjukkan sekumpulan elemen seperti kelas beserta isi dan hubungannya. Notasi yang digunakan dalam C*lass Diagram* dapat dilihat pada Gambar 2.5 di bawah ini.



Gambar 2.6 Notasi C*lass Diagram*

(Sumber: Rumbaugh, Jacobson, dan Booch, 1999, p44)