

BAB II

TINJAUAN PUSTAKA

1.1 Konsep Sistem Informasi

1.1.1 Pengertian Sistem

Sistem adalah kumpulan dari komponen-komponen yang saling berhubungan dan saling bekerja sama sebagai satu kesatuan organisasi untuk mencapai suatu tujuan yang sama serta dapat mempengaruhi sebagian yang akan mempengaruhi keseluruhan [8].

Model umum sebuah sistem adalah *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu, sebuah sistem memiliki karakteristik atau sifat-sifat tertentu yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem [8].

Adapun karakteristik yang dimaksud adalah [9] :

1. Komponen Sistem (*Components*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk sub-sistem.

2. Batasan Sistem (*Boundary*)

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem (*Environment*)

Bentuk apa pun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar sistem yang menguntungkan merupakan energi bagi sistem tersebut. Dengan demikian lingkungan luar tersebut harus dijaga dan dipelihara

Lingkungan luar yang merugikan harus dikendalikan kalau tidak, maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem (*Interface*)

Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan proses pengaliran sumber daya dari satu subsistem ke subsistem lainnya.

5. Masukan Sistem (*Input*)

Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contoh, di dalam suatu unit sistem komputer. "Program" adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan "data" adalah sinyal *input* untuk diolah menjadi informasi

6. Keluaran Sistem (*Output*)

Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi sub-sistem yang lain. Contoh sistem informasi keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang menjadi *input* bagi sub-sistem lain

7. Pengolah Sistem (*Process*)

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akuntansi Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen

8. Sasaran Sistem (*Objective*)

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Jika suatu sistem tidak memiliki sasaran maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

Sistem memiliki ciri-ciri yang dapat diklasifikasikan sebagai berikut [10] :

- a. *Component* : Suatu sistem harus memiliki beberapa elemen atau unsur-unsur atau unit-unit yang tersendiri namun dengan sistem tersebut, seperti paru-paru dalam sistem pernafasan

- b. *Boundary* : Batas suatu sistem tentunya harus berbeda atau terpisah dengan sistem lain atau lingkungan diluar sistem
- c. *Environment* : Lingkungan luar, sisi/bagian yang bukan termasuk kedalam suatu sistem
- d. *Interface* : *Connector*/penghubung antar *element* luar dengan sistem *input* masukan yang akan diproses oleh sistem
- e. *Process* : Pengolah, sistem harus memiliki unit pengolahan.
- f. *Output* : Keluaran atau hasil dari pengolahan
- g. *Objective* : Suatu sistem memiliki sasaran atau tujuan (*goal*).

1.1.2 Pengertian Informasi

Informasi merupakan suatu data yang telah diolah, diklasifikasikan dan diinterpretasikan serta digunakan untuk proses pengambilan keputusan [11].

1. Pengelompokan Informasi

a. Informasi Strategis

Informasi ini digunakan untuk mengambil keputusan jangka panjang, yang mencakup informasi eksternal, rencana perluasan perencanaan, dan sebagainya.

b. Informasi Taktis

Informasi ini dibutuhkan untuk mengambil keputusan jangka menengah, seperti informasi tren penjualan yang dapat dimanfaatkan untuk menyusun rencana penjualan.

c. Informasi Teknis

Informasi ini dibutuhkan untuk keperluan operasional sehari – hari, seperti informasi persediaan *stock*, retur penjualan, dan laporan kas harian.

2. Karakteristik Informasi

a. Relevan

Informasi harus memiliki makna yang tinggi sehingga tidak menimbulkan keraguan bagi yang menggunakannya dan dapat digunakan secara tepat untuk membuat keputusan.

b. Andal

Suatu informasi harus memiliki keterandalan yang tinggi, informasi yang dijadikan alat pengambilan keputusan merupakan kejadian nyata dalam aktivitas

perusahaan.

c. Lengkap

Informasi tersebut harus memiliki penjelasan yang rinci dan jelas dari setiap aspek peristiwa yang diukurnya.

d. Tepat Waktu

Setiap informasi harus dalam kondisi yang *update* tidak dalam bentuk yang usang, sehingga penting untuk digunakan sebagai pengambilan keputusan.

e. Dapat Dipahami

Informasi yang disajikan dalam bentuk yang jelas akan memudahkan orang dalam menginterpretasikannya.

3. Kualitas Informasi

a. Akurat (*accurate*)

Informasi harus bebas dari kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti bahwa informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi mungkin banyak mengalami gangguan (*noise*) yang dapat mengubah atau merusak informasi tersebut.

b. Tepat waktu (*timelines*)

Informasi yang sampai kepada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi, karena informasi merupakan landasan di dalam pengambilan keputusan. Bila pengambilan keputusan terlambat maka dapat berakibat fatal bagi organisasi.

c. Relevan (*relevance*)

Informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk setiap orang berbeda. Menyampaikan informasi tentang penyebab kerusakan mesin produksi kepada akuntan perusahaan tentunya kurang relevan. Akan lebih relevan bila ditujukan kepada ahli teknik perusahaan.

2.1.3 Sistem Informasi

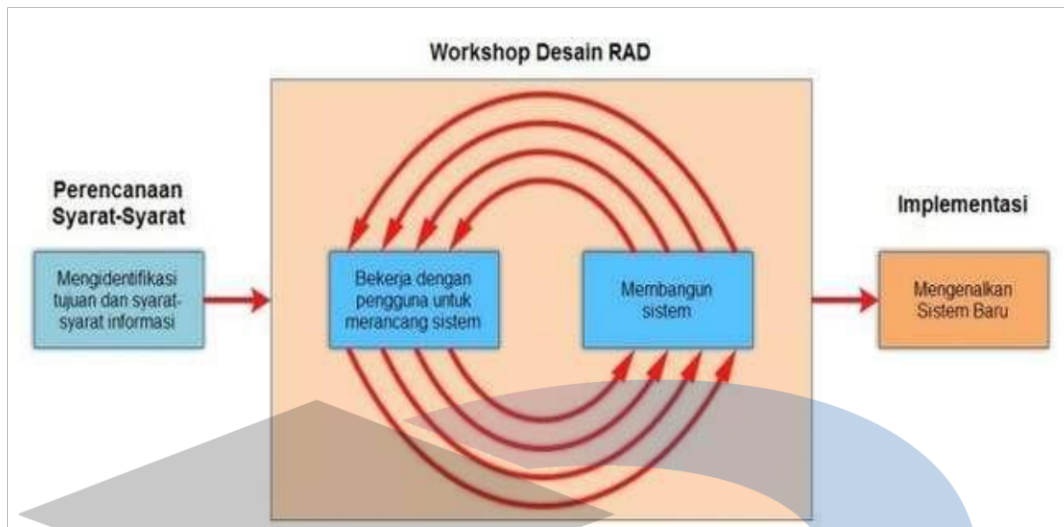
Sistem informasi adalah sekumpulan komponen yang saling berhubungan mengumpulkan atau mendapatkan memproses, menyimpan, dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan dalam suatu organisasi [12].

Komponen-komponen dari sistem informasi adalah sebagai berikut [13]:

1. Komponen *input* adalah data yang masuk kedalam sistem informasi.
2. Komponen model adalah kombinasi prosedur, logika dan model matematika yang memproses data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan keluaran yang diinginkan.
3. Komponen *output* adalah hasil informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.
4. Komponen teknologi adalah alat dalam sistem informasi, teknologi digunakan dalam menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan *output* dan memantau pengendalian sistem
5. Komponen basis data adalah kumpulan data yang saling berhubungan yang tersimpan didalam komputer dengan menggunakan *software database*.
6. Komponen *control* adalah komponen yang mengendalikan gangguan terhadap sistem informasi.

2.2 Rapid Application Development (RAD)

RAD merupakan model proses perangkat lunak yang menekankan pada daur pengembangan hidup yang singkat. *RAD* merupakan versi adaptasi cepat dari model *waterfall*, dengan menggunakan pendekatan konstruksi komponen. *RAD* merupakan gabungan dari bermacam-macam teknik terstruktur dengan teknik *prototyping* dan mempercepat pengembangan teknik pengembangan *joint application* untuk sistem/aplikasi. Dari definisi konsep *RAD* ini, dapat dilihat bahwa pengembangan aplikasi dengan menggunakan metode *RAD* dapat dilakukan dalam waktu yang relatif lebih cepat. Sesuai dengan metodologi *RAD* berikut ini adalah tahap-tahap pengembangan aplikasi dari tiap-tiap fase pengembangan aplikasi dapat di lihat pada gambar dibawah ini [14].



Gambar 2.1 Tahap Pengembangan RAD

Tahapan RAD terdiri dari 3 tahap yang terstruktur dan saling bergantung di setiap tahap, yaitu [14] :

1. *Requirements Planning* (Perencanaan Persyaratan).
 - a. Pengguna dan analisis bertemu untuk mengidentifikasi tujuan dari aplikasi atau sistem.
 - b. Berorientasi pada pemecahan masalah bisnis
2. *Design Workshop*
 - a. Fase design dan menyempurnakan.
 - b. Gunakan kelompok pendukung keputusan sistem untuk membantu pengguna setuju pada design
 - c. *Programmer* dan analis membangun dan menunjukkan tampilan visual C, design dan alur kerja pengguna
 - d. Pengguna menanggapi *prototype* kerja aktual
 - e. Analisis menyempurnakan modul dirancang berdasarkan tanggapan pengguna
3. *Implementation* (Penerapan)
 - a. Sebagai sistem yang baru dibangun, sistem baru atau parsial diuji dan diperkenalkan kepada organisasi.
 - b. Ketika membuat sistem baru, tidak perlu untuk menjalankan sistem yang lama secara paralel.

2.3 Use Case Diagram

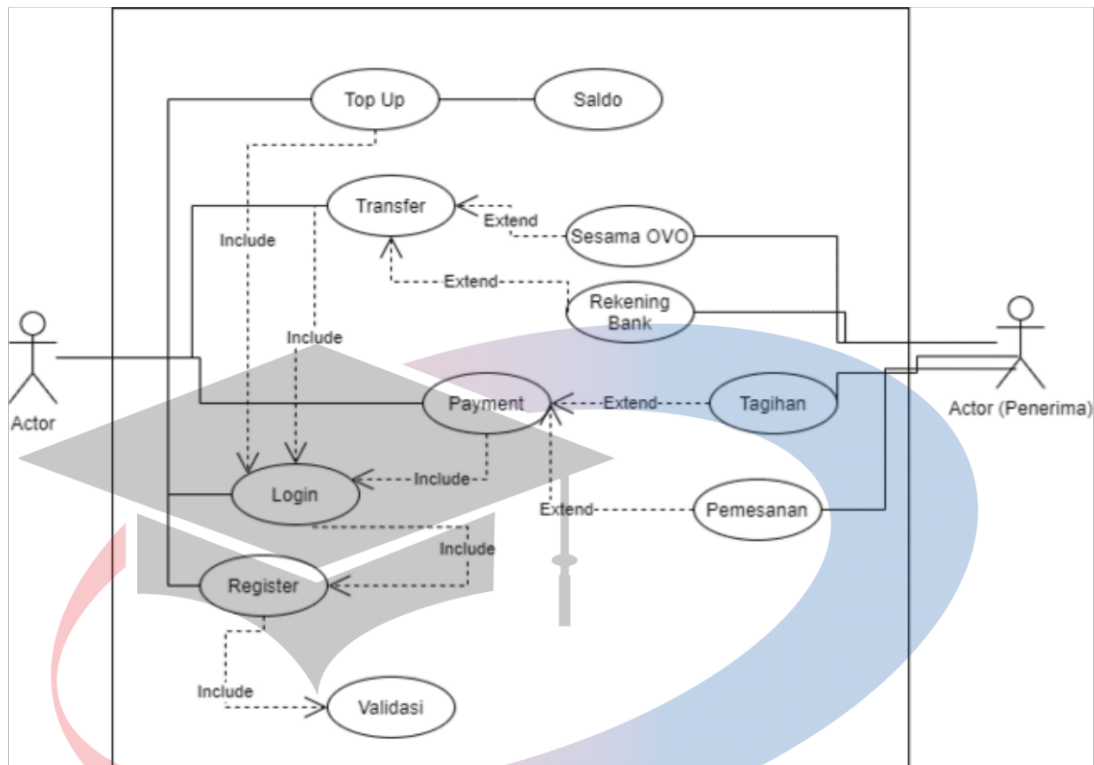
Suatu *use case* diawali dengan memasukkan *input* dari seorang pemakai *use case* merupakan suatu kejadian-kejadian yang diajukan oleh seorang pemakai serta spesifikasi interaksi antara pemakai dengan sistem *use case* yang sederhana hanya melibatkan beberapa interaksi/hubungan dengan sebuah pemakai, dan *use case* yang lebih kompleks melibatkan beberapa interaksi dengan pemakai *use case* yang lebih kompleks juga melibatkan lebih dari satu aktor. Untuk menjelaskan *use case* dalam sistem, sangat bagus bila diawali dengan memperhatikan pemakai dan aksi yang dilakukan didalam sistem. Setiap *use case* menggambarkan suatu urutan interaksi antara pengguna dengan sistem itu [15].

Tabel 2.1 Simbol-Simbol Dasar *Use Case Diagram*

No	Simbol	Nama	Keterangan
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
3		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek

			induk (<i>ancestor</i>).
4		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya..
7		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
8		<i>Sistem</i>	Mendeskripsikan paket-paket yang secara terbatas.
9		<i>Use Case</i>	Deskripsi yang berasal dari barisan aksi-aksi yang akan menampilkan sistem yang membentuk sesuatu yang akan terjadi pada <i>actor</i> .

Berikut ini akan dijelaskan contoh *use case diagram* dan juga penjelasannya [16]:



Gambar 2.2 Contoh *Use Case Diagram* pada OVO

Adapun penjelasannya sebagai berikut [16]:

1. *User* : Orang yang dapat mengakses atau menggunakan aplikasi OVO, mulai dari *login* ke aplikasi hingga melakukan aksi terhadap aplikasi seperti *top up* saldo, *transfer*, dan *payment*.
2. *Register* : *Register* merupakan langkah pertama yang dilakukan *user* ketika ia tidak mempunyai akses pada aplikasi OVO. Mendaftarkan data diri ke dalam aplikasi agar dikenali.
3. *Login* : Setelah mendapatkan akun, *user* harus melakukan *login* agar dapat mengakses berbagai fitur aplikasi OVO.
4. *Top up* : Suatu kegiatan yang dilakukan *user* untuk mengisi ulang saldo OVO. Terdapat 2 pilihan alternatif untuk melakukan *top up* saldo, yaitu melalui ATM dan internet *banking*.
5. *Transfer* : *Transfer* berfungsi untuk mengirim atau membagikan saldo dalam aplikasi OVO ke pengguna lain baik sesama OVO atau ke rekening tertentu.

6. *Payment* : Ketika *user* memilih menu *payment*, maka *user* dapat melakukan pembayaran lewat aplikasi.

Use case description berfungsi melengkapi diagram *use case* agar lebih dipahami konteksnya. Disajikan dalam bentuk tabel, *use case description* adalah gambaran secara general mengenai fungsionalitas proses bisnis berupa skenario yang melibatkan berjalannya suatu sistem. Ada 13 komponen yang terdapat pada *use case description* untuk menjelaskan masing-masing *use case* secara lengkap. Berikut adalah penjelasan dari komponen yang terdapat pada *use case description* tersebut [17].

Tabel 2.2 Komponen Pada *Use Case Description*

No	Komponen	Deskripsi
1	<i>Use Case ID:</i>	Berisikan kode unik untuk membedakan setiap <i>use case</i>
2	<i>Use Case Name</i>	Berisi nama <i>use case</i> .
3	<i>Description</i>	Berisi penjelasan singkat mengenai fungsi <i>use case</i>
4	<i>Actor</i>	Berisikan informasi mengenai pengguna atau aktor yang terlibat pada <i>use case</i> tersebut.
5	<i>Pre Condition</i>	Berisikan kondisi yang harus ada atau sudah terjadi sebelum <i>use case</i> dijalankan. Prasyarat apa yang harus dipenuhi sebelum <i>use case</i> dijalankan.
6	<i>Post Condition</i>	Proses yang dihasilkan dari kegiatan <i>use case</i> .
7	<i>Basic Flow</i>	Berisi penjelasan langkah-langkah normal yang berakhir sukses. Ditandai dengan awal, <i>body</i> dan akhir
8	<i>Exceptions Flow</i>	Menjelaskan informasi mengenai kendala-kendala yang

		menyebabkan skenario dasar tidak dapat dipenuhi.
9	<i>Variations</i>	Berisi mengenai tindakan alternatif apabila ada perkecualian dari skenario dasar. Merupakan representasi notasi <i><extend></i> pada diagram <i>use case</i> .
10	<i>Extensions</i>	Berisi mengenai skenario tambahan yang merujuk pada notasi <i><include></i> dan biasanya diidentifikasi apabila ada langkah-langkah yang terlibat tetapi tidak terkait dengan arus konteks skenario awal.
11	<i>Business Rules</i>	Berisi tentang aturan bisnis yang terdapat pada <i>use case</i> .
12	<i>Non Functional Requirement</i>	Penyampaian kebutuhan diluar fungsi <i>use case</i> dapat disampaikan disini. Umumnya berisi kinerja, kapasitas dan keterbatasan sistem dan lain-lain

Adapun contoh dari *use case description* adalah [17]:

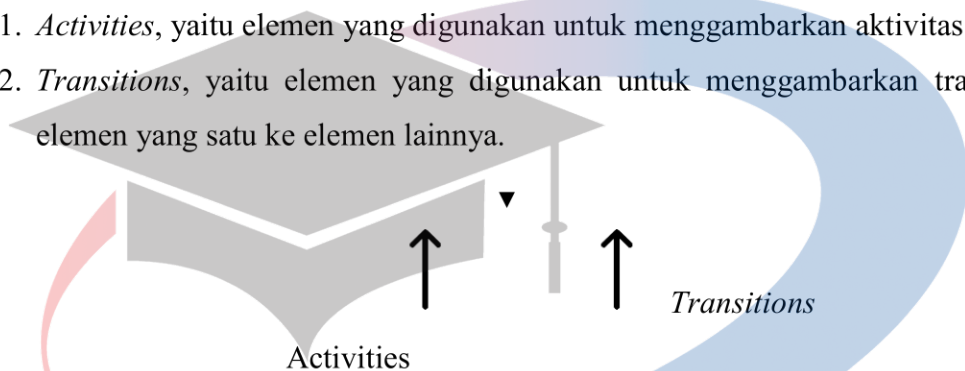
Use Case ID :		
Use Case Name :		
Description :		
Actor :		
Pre Condition :		
Post Condition :		
	Actors	System
Basic Flow	1.	2.
	3.	4.
Variations :		
Exceptions :		
Extensions :		
Business Rules :		
Non Functional Req.:		

Gambar 2.3 Contoh Tampilan Pada *Use Case Description*

2.4 Activity Diagram

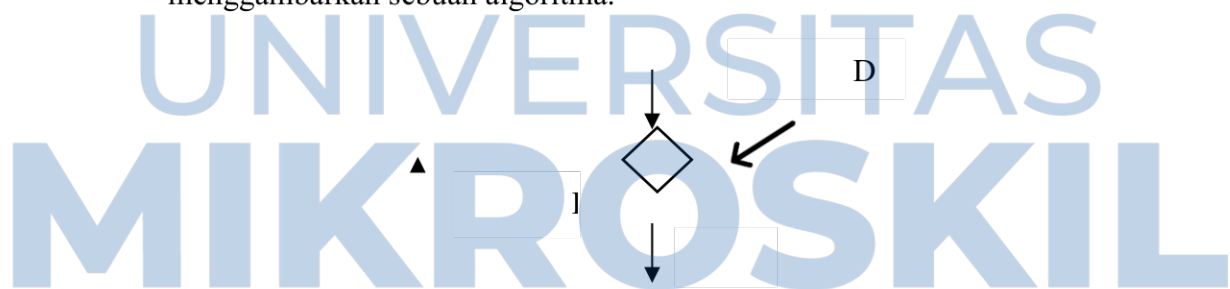
Activity diagram, yaitu diagram yang digunakan untuk menggambarkan alur kerja (aktivitas) pada *use case* (proses), logika, proses bisnis dan hubungan antara aktor dengan alur-alur kerja *use case*. Jika anda sudah terbiasa dengan *flowchart*, maka anda tidak akan merasa kesulitan dalam mempelajari *activity diagram*, karena *activity diagram* identik dengan *flowchart*, hanya saja ada beberapa notasi tambahan yang digunakan kasus-kasus tertentu. Berikut elemen-elemen dari *activity diagram* [18]:

1. *Activities*, yaitu elemen yang digunakan untuk menggambarkan aktivitas.
2. *Transitions*, yaitu elemen yang digunakan untuk menggambarkan transisi dari elemen yang satu ke elemen lainnya.



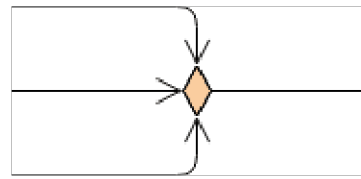
Gambar 2.4 Contoh Elemen *Activities* dan *Transitions*

3. *Decisions*, yaitu elemen yang digunakan untuk percabangan logika elemen ini sering dijumpai pada *flowchart* terutama *flowchart* yang digunakan untuk menggambarkan sebuah algoritma.



Gambar 2.5 Contoh Elemen *Decisions*

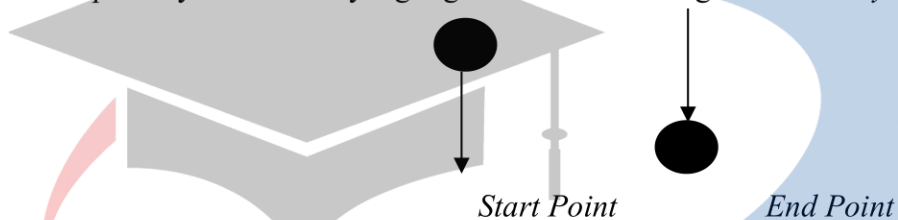
4. *Merge point*, yaitu elemen yang digunakan untuk menggabungkan percabangan yaitu elemen yang digunakan untuk menggabungkan percabangan proses. Elemen ini merupakan kebalikan dari elemen *decisions*, dimana jika *decisions* digunakan untuk percabangan, sedangkan *merge point* digunakan sebagai penggabungan dari percabangan.



Decisions

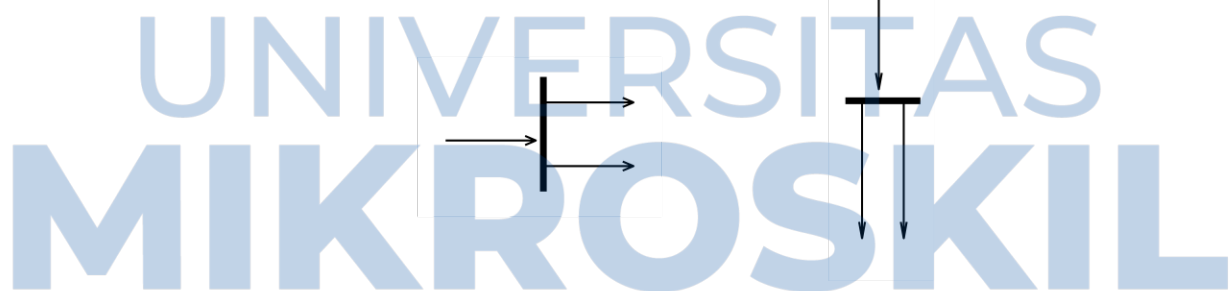
Gambar 2.6 Contoh Notasi Merge Point

5. *Start point*, yaitu elemen yang digunakan untuk memulai *activity diagram*
6. *End point*, yaitu elemen yang digunakan untuk mengakhiri *activity diagram*



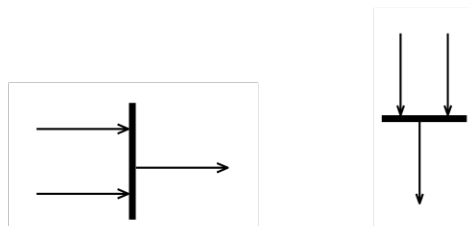
Gambar 2.7 Contoh Elemen *Start Point* dan *End Point*

7. *Concurrency*, yaitu elemen yang digunakan sebagai percabangan proses (bukan percabangan logika). Proses yang ada di dalam elemen ini, bisa dilakukan secara *random* (tidak berurutan).



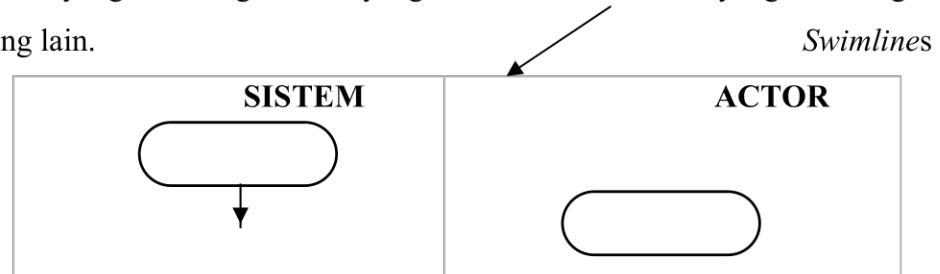
Gambar 2.8 Contoh Notasi *Concurrency*

8. *Synchronization*, yaitu elemen yang digunakan untuk menggabungkan proses yang dipisahkan oleh *concurrency*



Gambar 2.9 Contoh Elemen *Synchronization*

9. *Swimline*, yaitu elemen yang digunakan untuk memisahkan antara *actor* dan sistem ataupun *actor* yang satu dengan *actor* yang lain atau antara sistem yang satu dengan sistem yang lain.



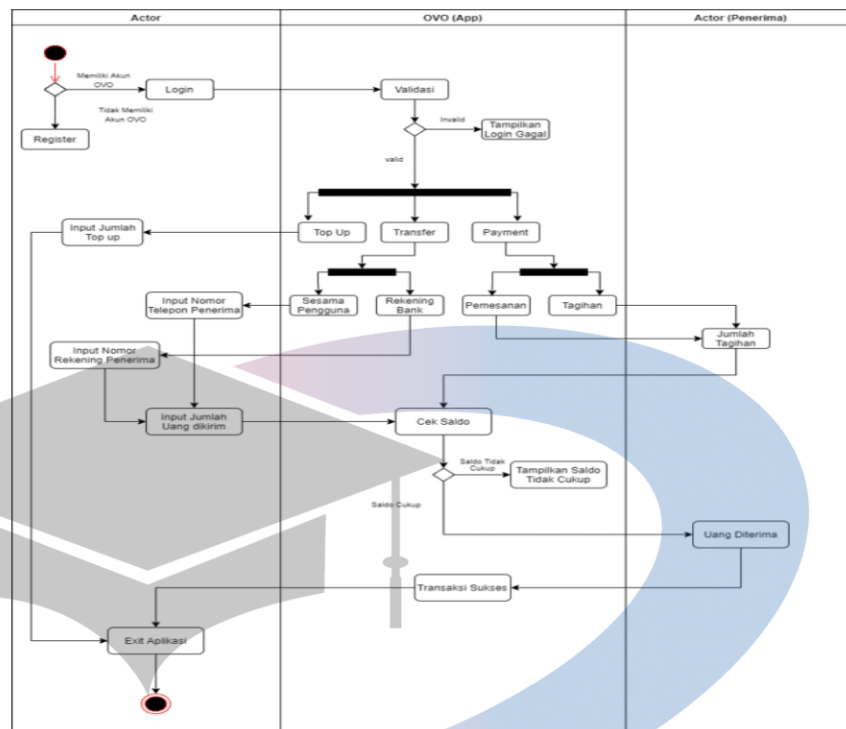
Gambar 2.10 Contoh Elemen *Swimline*

Berikut ini merupakan simbol *activity diagram* dan contoh kasus dari penggunaan *activity diagram*. Kasus yang diambil seputar cara penggunaan OVO sebagaimana yang sudah dibuatkan *use case* modelnya diatas [19].

Tabel 2.3 Simbol *Activity Diagram*

No	Gambar	Nama	Keterangan
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		<i>Action</i>	<i>State</i> dari sistem yang mencerminkan eksekusi dari suatu aksi
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan dihancurkan
5		<i>Extend</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Dibawah ini akan dijelaskan contoh dari *activity diagram* beserta penjelasannya [20]:



Gambar 2.11 Contoh *Activity Diagram* Penggunaan OVO

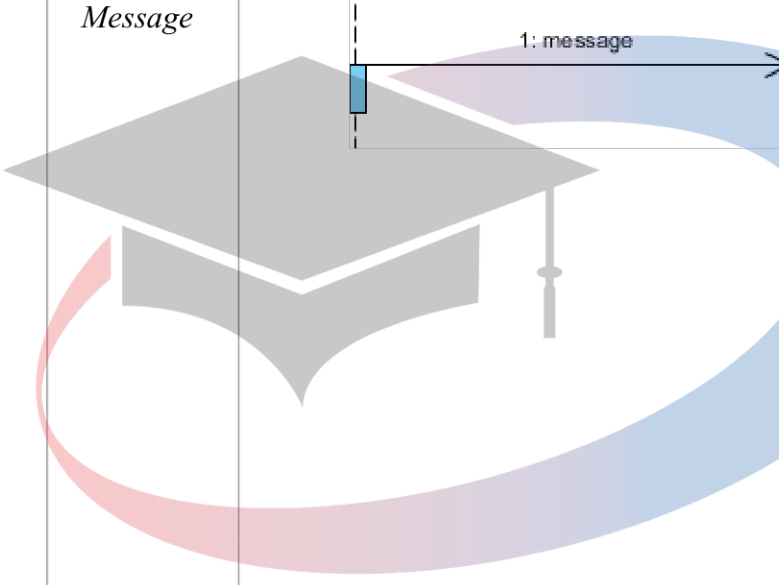
Activity diagram ini memberikan informasi kepada kita tentang prosedur penggunaan OVO. Mulai dari top up, cek saldo, transfer ke sesama pengguna OVO hingga membayar tagihan. Disini kita dapat melihat fungsi dari masing-masing elemen yang sudah digambarkan diatas [16].

2.5 *Sequence Diagram*

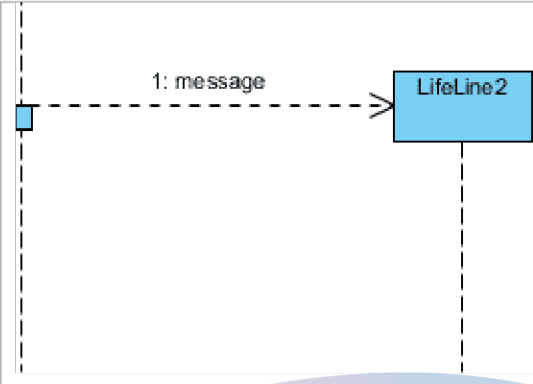
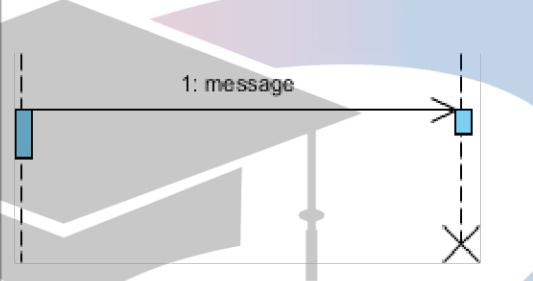
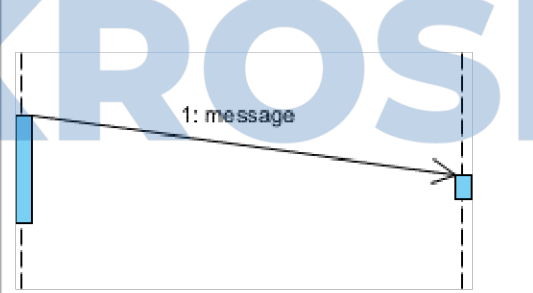
Sequence diagram adalah *tools* yang sangat populer dalam pengembangan sistem informasi secara *object-oriented* untuk menampilkan interaksi antar objek. Berdasarkan definisi tersebut, dapat disimpulkan bahwa *sequence diagram* adalah *tools* yang digunakan dalam pengembangan sistem [18].


Tabel 2.4 Simbol-Simbol Dasar *Sequence Diagram*

No	Nama	Simbol	Keterangan
1	<i>Actor</i>		<p>Jenis peran yang dimainkan oleh entitas yang berinteraksi dengan subjek (Misalnya dengan bertukar sinyal dan data)</p> <p>Di luar subjek (maksudnya adalah batasan/kendala dari aktor bukan bagian dari batasan/kendala dari subjek yang sesuai).</p> <p>Mewakili peran yang dimainkan oleh pengguna manusia, perangkat keras eksternal, atau subjek lainnya.</p>
2	<i>Lifeline</i>		<p><i>Lifeline</i> atau Garis hidup mewakili peserta individu dalam Interaksi.</p>
3	<i>Activation Box</i>		<p><i>Activation box</i> atau kotak aktivasi berbentuk sebuah persegi panjang tipis pada <i>lifeline</i>, mewakili periode di mana suatu elemen melakukan operasi.</p>

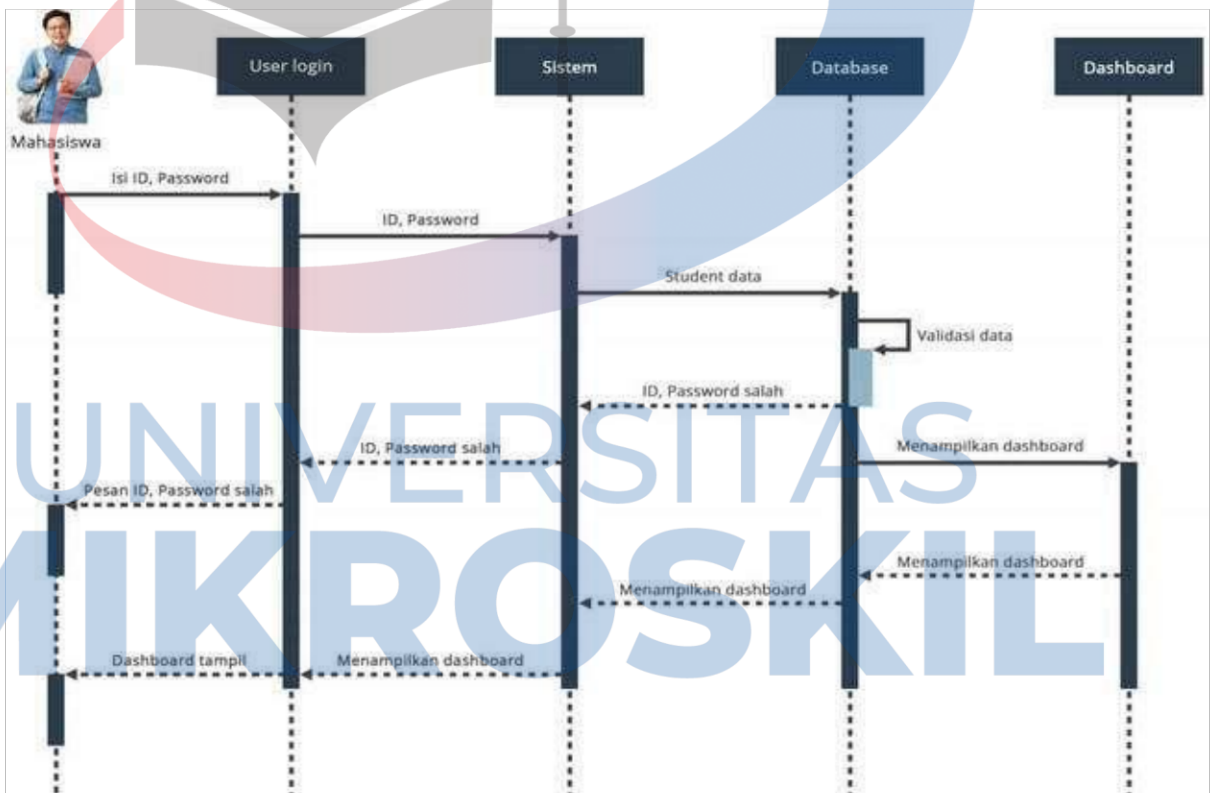
			Bagian atas dan bawah dari kotak aktivasi disejajarkan dengan inisiasi dan waktu penyelesaian masing-masing.
4	<i>Call Message</i>		<p><i>Call message</i> atau pesan panggilan merupakan sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Call Message</i> adalah jenis pesan yang mewakili permintaan operasi dari target <i>lifeline</i>.</p>
5	<i>Return Message</i>		<p><i>Return message</i> atau pesan balik adalah sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Return message</i> adalah jenis pesan yang mewakili informasi yang dikirimkan kembali ke pengirim pesan atau pemanggil berdasarkan pesan sebelumnya.</p>

6	<i>Self Message</i>		<p><i>Self message</i> atau pesan mandiri adalah sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Self message</i> adalah jenis pesan yang mewakili permohonan pesan dari <i>lifeline</i> yang sama</p>
7	<i>Recursive Message</i>		<p><i>Recursive Message</i> atau pesan rekursif merupakan sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Recursive Message</i> merupakan jenis pesan yang mewakili permohonan pesan dari <i>lifeline</i> yang sama. <i>Recursive Message</i> ini menargetkan aktivasi di atas aktivasi tempat pesan itu berasal.</p>
8	<i>Create Message</i>		<p><i>Create Message</i> merupakan sebuah pesan yang mendefinisikan komunikasi tertentu antara</p>

			<p><i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Create message</i> atau membuat pesan adalah jenis pesan yang mewakili instansiasi (target) <i>lifeline</i>.</p>
9	<i>Destroy Message</i>		<p><i>Destroy Message</i> merupakan sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p><i>Destroy message</i> atau hancurkan pesan merupakan sebuah jenis pesan yang mewakili permintaan untuk menghancurkan siklus hidup target.</p>
10	<i>Duration Message</i>		<p><i>Duration message</i> atau pesan durasi yaitu sebuah pesan yang mendefinisikan komunikasi tertentu antara <i>Lifelines</i> dari sebuah interaksi.</p> <p>Pesan Durasi menunjukkan jarak antara dua batasan waktu untuk permohonan pesan</p>

11	Note		<i>Note</i> atau Catatan (komentar) merupakan catatan atau lampiran berbagai komentar ke elemen. Sebuah komentar tidak memiliki kekuatan semantik, tetapi dapat berisi informasi yang berguna bagi pemodal.
----	------	---	---

Berikut ini adalah contoh dari *sequence diagram* beserta penjelasannya [18]:



Gambar 2.12 Contoh *Sequence Diagram*

Sequence diagram diatas terdapat satu aktor (mahasiswa) dan empat objek, yaitu *user login*, *sistem*, *database*, dan *dashboard*. Pertama-tama mahasiswa akan masuk ke tampilan *user login* dengan menggunakan ID dan *Password*. Lalu, sistem akan mengirimkan data tersebut ke *database* untuk divalidasi. Di dalam *database* data mahasiswa akan diperiksa dan divalidasi. Jika data yang dimasukan salah dan tidak

valid, maka akan menampilkan pesan bahwa ID atau *Password* salah, sedangkan jika data yang dimasukkan benar dan valid, maka sistem akan menampilkan *dashboard* aplikasi [18].


2.6 Class Diagram


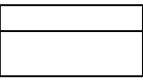


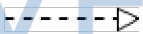
Class Diagram adalah ilustrasi hubungan antara *class* yang dimodelkan dalam sistem *class diagram* dimana sangat mirip dengan diagram hubungan entitas [20]. *Class Diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class Diagram* secara khas meliputi: Kelas (*Class*), Relasi *Associations*, *Generalization* dan *Aggregation*, atribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality* [21].

Tabel 2.5 Multiplicity *Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Minimal 2 maksimal 4

Tabel 2.6 Simbol *Class Diagram*

No	Gambar	Nama	Keterangan
1		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

2		<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
6		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.

Class menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Sebuah *class diagram* terdiri dari sejumlah kelas yang dihubungkan dengan garis yang menunjukkan hubungan antar kelas yang disebut dengan *Associations* [22].

Simbol	Keterangan
	<p><i>Class</i></p> <p>1. <i>class name</i></p> <p>2. <i>attributes</i></p> <p>3. <i>behaviors</i></p>
	<i>Association!</i>
	<i>Agregation</i>
	<i>Generalization</i>

Gambar 2.13 Notasi *Class Diagram*

2.7 Pengertian Aplikasi *Mobile*

Aplikasi *mobile* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat diartikan sebagai perpindahan dari suatu tempat ke tempat yang lain. Kata *mobile* mempunyai arti bergerak atau berpindah, sehingga aplikasi *mobile* adalah sebutan untuk aplikasi yang berjalan di *mobile device*. Dengan menggunakan aplikasi *mobile*, dapat dengan mudah melakukan berbagai macam aktivitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya. Pemanfaatan aplikasi *mobile* untuk hiburan paling banyak digemari oleh pengguna telepon seluler, karena dengan memanfaatkan adanya fitur *game*, *music player*, sampai *video player* membuat kita menjadi semakin mudah menikmati hiburan kapan saja dan dimanapun. Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, design *layout*, tetapi mereka memiliki kesamaan karakteristik yang sangat berbeda dari *desktop* sistem. Perangkat *mobile* memiliki *memory* yang kecil [23].

Secara umum perangkat *mobile* memiliki karakteristik [24]:

1. *Central processing unit* (CPU) dan *Graphical Processor Unit* (GPU) yang terbatas,

2. Layar yang kecil,
3. Lingkungan kerja yang beragam (*mobile context*), dan
4. Koneksi jaringan yang tidak *reliable*

2.8 Pengertian Arsitek

Arsitek didefinisikan sebagai seorang “perancang bangunan” (*building designer*), namun peran arsitek tidak hanya sebatas bangunan saja, tetapi meliputi tugas penataan (penciptaan dan perwujudan) dari ruang dalam skala yang lebih luas. Ruang tersebut berwujud lingkungan binaan (*build environment*) yang diperuntukkan bagi kehidupan manusia maupun masyarakat luas (umum). Dalam skala kecil (mikro) tugas dan peran arsitek adalah menata ruangan-ruangan (*rooms*) yang diintegrasikan secara utuh dalam bentuk bangunan (*building*). Dalam skala mikro ini, arsitek menjalankan tugasnya sebagai “perancang bangunan” (*building designer*). Seorang arsitek akan berupaya secara maksimum dalam proses menciptakan bangunan, dimana digunakan kaidah-kaidah pedoman-pedoman dalam perancangan arsitektur. Pemenuhan tujuan utama arsitektur seperti [25]:

1. Pemenuhan aspek fungsi/kegunaan bangunan
2. Pemenuhan aspek struktur/kekuatan bangunan hingga
3. Pemenuhan aspek keindahan bangunan. Dalam skala perancangan bangunan, pemahaman “tugas dari bangunan” (*the building task*) menjadi penting bagi seorang arsitek/perancang bangunan. Demikian pula pemahaman terhadap aspek “keteknikan bangunan” (*the building technique*) merupakan tugas yang mesti dilakukan dan diselesaikan. Tugas selanjutnya yaitu melakukan kreasi dalam “mengekspresikan bentuk” bangunan sebagai bagian dari pencapaian unsur estetika/keindahan bangunan. Dalam skala yang lebih luas, tugas dari seorang arsitek bukan lagi (hanya) menciptakan dan mewujudkan bangunan, tetapi lebih luas dari itu menyangkut di dalamnya aspek tapak dan lingkungan sekitarnya (*site and surrounding*). Bahkan arsitek perlu mengenal, mengerti dan memahami aspek-aspek yang berkaitan dengan penataan lingkungan dan penataan ruang. Oleh karena itu dalam skala makro, tugas seorang arsitek juga berkaitan setidaknya dengan tiga tingkatan:
 - a. Penataan/tata bangunan

- b. Penataan/tata lingkungan dan
- c. Penataan/tata ruang.

Secara kerangka kerja keprofesian, maka tugas dan peran dari seorang arsitek akan berhubungan terutama dengan [25]:

1. *Interior designer* dan *furniture designer* (dalam skala mikro)
2. *Structural engineer, mechanical & electrical engineer* (dalam skala *middle*) serta, *planolog/urban planner, urban designer* dan arsitek lanskap (dalam skala makro)



UNIVERSITAS MIKROSKIL