BAB II
KAJIAN LITERATUR

2.1 Konsep Sistem Informasi

Sistem didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama - sama, Secara garis besar, sebuah sistem In*form*asi terdiri atas tiga komponen utama. Ketiga komponen tersebut mencakup *software, hardware*, dan *brainware* [4].

Software mencakup semua perangkat lunak yang dibangun dengan bahasa pemrograman tertentu, pustaka, untuk kemudian menjadi sistem operasi, aplikasi, dan *driver*. Sistem operasi, aplikasi, *driver*, saling bekerja sama agar komputer dapat berjalan dengan baik. *Hardware* mencakup semua perangkat keras (*motherboard*, *processor, VGA*, dan lainnya) yang disatukan menjadi sebuah komputer. Dalam konteks yang luas, bukan hanya sebuah komputer, namun sebuah jaringan komputer. *Brainware* mencakup kemampuan otak manusia, yang mencakup ide, pemikiran, analisis, di dalam menciptakan dan menggabungkan *hardware* dan *software* [4].

In*form*asi merupakan hasil pengolahan data dari satu atau berbagai sumber, yang kemudian diolah, sehingga memberikan nilai, arti, dan manfaat. Proses pengelolaan ini memerlukan teknologi [4].

Pada proses pengolahan data, untuk dapat menghasilkan In*form*asi, juga dilakukan proses verifikasi secara akurat, spesifik, dan tepat waktu. Hal ini penting agar In*form*asi dapat memberikan nilai dan pemahaman kepada pengguna. Pengguna dalam hal ini mencakup pembaca, pendengar, penonton, bergantung pada bagaimana cara pengguna tersebut menikmati sajian In*form*asi dan melalui media apa In*form*asi tersebut disajikan [4].

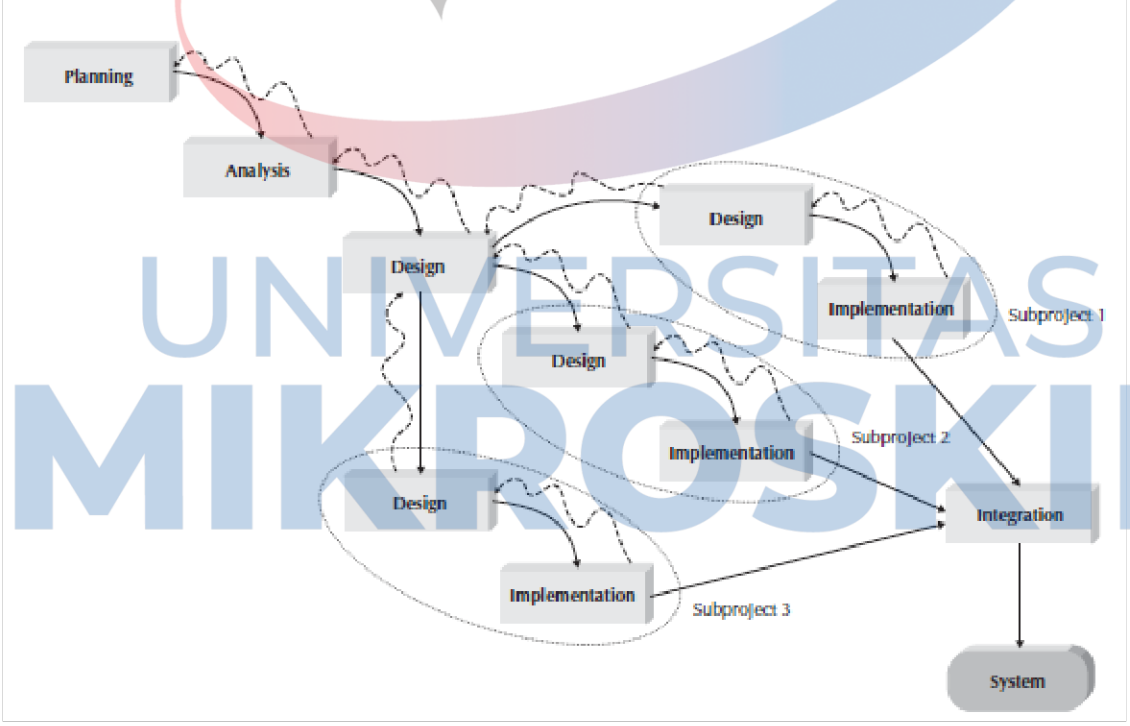
Sistem In*form*asi merupakan gabungan dari empat bagian utama. Keempat bagian utama tersebut mencakun perangkat lunak (*software*), perangkat keras (*hardware*), infrastruktur, dan Sumber Daya Manusia (SDM) yang terlatih. Keempat bagian utama ini saling berkaitan untuk menciptakan sebuah sistem yang dapat mengolah data menjadi In*form*asi yang bermanfaat [4].

Sistem In*form*asi memiliki beberapa komponen dan beberapa elemen, yang mana antar komponen dan antar-elemen ini saling bekerja sama, saling terkait, dan memiliki fungsional kerja yang menyatu, sehingga sistem In*form*asi dapat bekerja dengan baik [4].

Dalam penerapannya, sebuah sistem In*form*asi dapat berupa sebuah *mainframe*, sebuah server dari komputer biasa, maupun hosting di internet pada sebuah komputer server. Namun tetap saja ada kesamaaan di antara ketiga penerapan berbeda ini. Kesamaan itu yaitu sama-sama menggunakan sarana jaringan komputer (intranet maupun internet) untuk melakukan pemrosesan data secara bersama (terdistribusi), baik oleh beberapa pengguna maupun beberapa grup pengguna, menggunakan ayanan/fitur/aplikasi yang disertakan [4].

2.2 Rapid Application Development (RAD)

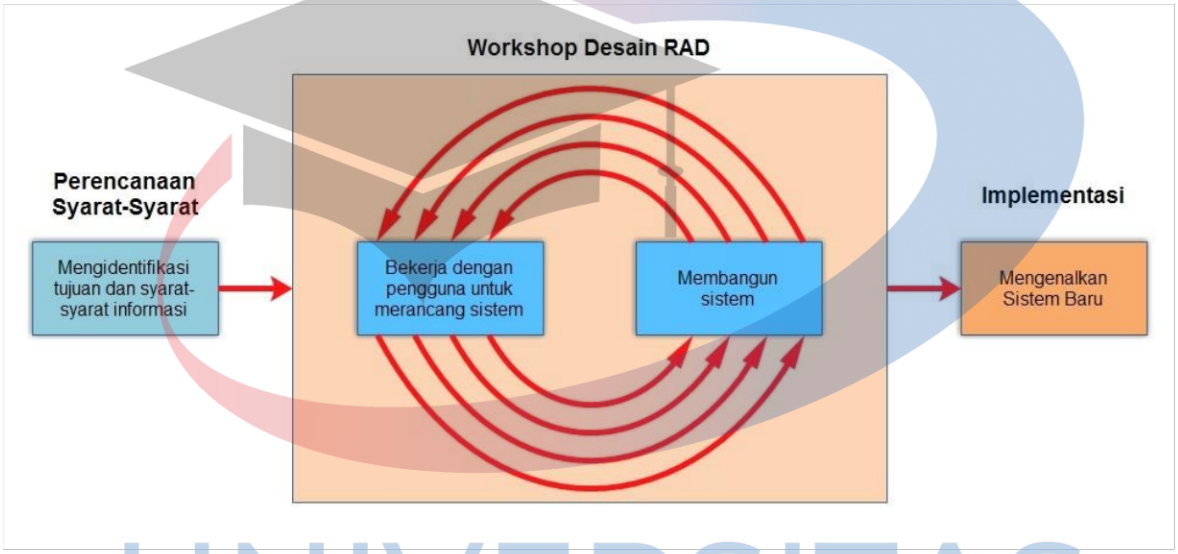
Metodologi perancangan sistem yang digunakan dalam penelitian ini yaitu *Rapid Application Development(RAD)*. Ini adalah kelas metodologi pengembangan sistem yang lebih baru yang muncul pada 1990-an. Metodologi berbasis RAD berusaha untuk mengatasi kedua kelemahan metodologi desain terstruktur dengan menyesuaikan fase SDLC agar beberapa bagian dari sistem dikembangkan dengan cepat dan sampai ke tangan pengguna. Dengan cara ini, pengguna dapat lebih memahami sistem dan menyarankan revisi yang membawa sistem lebih dekat dengan apa yang dibutuhkan [5].



Gambar 2.1 Metodologi Berbasis Pembangunan Paralel

Sebagian besar metodologi berbasis RAD merekomendasikan agar analis menggunakan teknik khusus dan alat komputer untuk mempercepat fase analisis, desain, dan implementasi, seperti alat rekayasa perangkat lunak berbantuan komputer (*CASE*), sesi desain aplikasi bersama (JAD), generasi keempat atau bahasa pemrograman visual yang

menyederhanakan dan mempercepat pemrograman, dan pembuat kode yang secara otomatis menghasilkan program dari spesifikasi desain. Kombinasi fase SDLC yang diubah dan penggunaan alat dan teknik ini meningkatkan kecepatan dan kualitas pengembangan sistem. Namun, ada satu kemungkinan masalah halus dengan metodologi berbasis RAD: mengelola ekspektasi pengguna. Karena penggunaan alat dan teknik yang dapat meningkatkan kecepatan dan kualitas pengembangan sistem, ekspektasi pengguna tentang apa yang mungkin dapat berubah secara dramatis. Sebagai pengguna yang lebih memahami teknologi In*form*asi (TI), persyaratan sistem cenderung berkembang. Ini tidak terlalu menjadi masalah saat menggunakan metodologi yang menghabiskan banyak waktu untuk mendokumentasikan persyaratan secara menyeluruh [5].



Gambar 2.2 Gambar Desain Workshop RAD

3 fase penyelesaian untuk restoran berbasis sistem In*form*asi manajemen metode RAD dapat dijelaskan sebagai berikut:[6]

1. *Requirements planning*

   Untuk tahap ini, peneliti bertemu dengan pemangku kepentingan untuk mengetahui keinginan dan mengidentifikasi tujuan sistem dan kebutuhan sistem In*form*asi untuk dibangun.

2. RAD *design workshop*

   Pada tahap ini, peneliti membangun dan menerapkan sistem yang dapat digambarkan sebagai sebuah bengkel. Peneliti membangun In*form*asi sistem dan langsung memvisualisasikan desain mereka dan proses bisnis kepada pemangku kepentingan. Selama proses lokakarya, pemangku kepentingan dapat segera merespon In*form*asi

tersebut sistem dibangun dan dapat memperbaiki modul yang ada bukan oleh keinginan pemangku kepentingan bisnis.

3. *Implementation*

   Pada fase ini, peneliti dan pemangku kepentingan saling berdiskusi untuk membuat sistem In*form*asi manajemen restoran selama *workshop*. Tidak hanya sistem tetapi juga merancang proses bisnis yang tepat untuk pemangku kepentingan sebelum digunakan di tempat usahanya. Setelah persetujuan dari semua aspek, sistem In*form*asi diuji dan diperkenalkan ke organisasi.

Kelebihan RAD: [6]

1. Produk yang dihasilkan akan lebih dapat diterima oleh konsumen.
2. Terdapat batasan-batasan pada sistem yang dibangun sehingga perubahan kebutuhan dapat ditampung
3. Siklus yang dilalui oleh perangkat lunak lebih pendek jika didukung dengan penggunaan alat-alat RAD yang kuat
4. Menghemat waktu, menghemat biaya, serta menghasilkan produk yang lebih berkualitas.

Kekurangan RAD: [6]

1. Tingkat manajemen yang lebih kompleks.
2. Pengguna harus terlibat dalam seluruh siklus pengembangan.
3. Dibutuhkan personil tim pengembang yang terampil di tiap bidangnya.
4. Sistem harus dapat dibagi menjadi modul-modul.

2.3 Alat Bantu Pengembangan Sistem

2.3.1 Use case Diagram

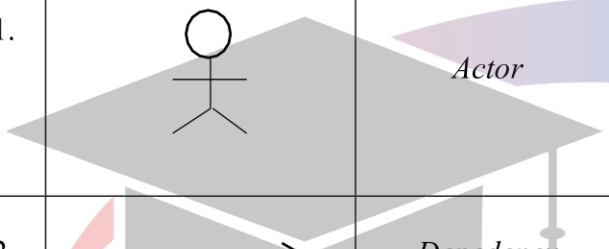
UML pada dasarnya didasarkan pada teknik analisis berorientasi objek yang dikenal dengan *use case* modeling. Model *use case* menunjukkan pandangan sistem dari perspektif pengguna, sehingga menjelaskan apa yang dilakukan oleh sistem tanpa menjelaskan bagaimana sistem melakukannya. UML dapat digunakan untuk menganalisis model *use case* dan menurunkan objek sistem beserta interaksi mereka dengan pengguna sistem. Dengan menggunakan teknik UML, Anda dapat menganalisis objek dan interaksi mereka untuk menurunkan perilaku, atribut, dan hubungan objek [7].

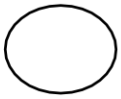
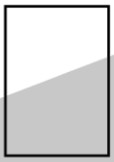
Use case memberikan pengembang pandangan tentang apa yang diinginkan oleh pengguna. *Use case* tidak terikat pada detail teknis atau implementasi. *Use case* dapat

dianggap sebagai urutan transaksi dalam sistem. Model *use case* didasarkan pada interaksi dan hubungan antara *use case* individu [7].

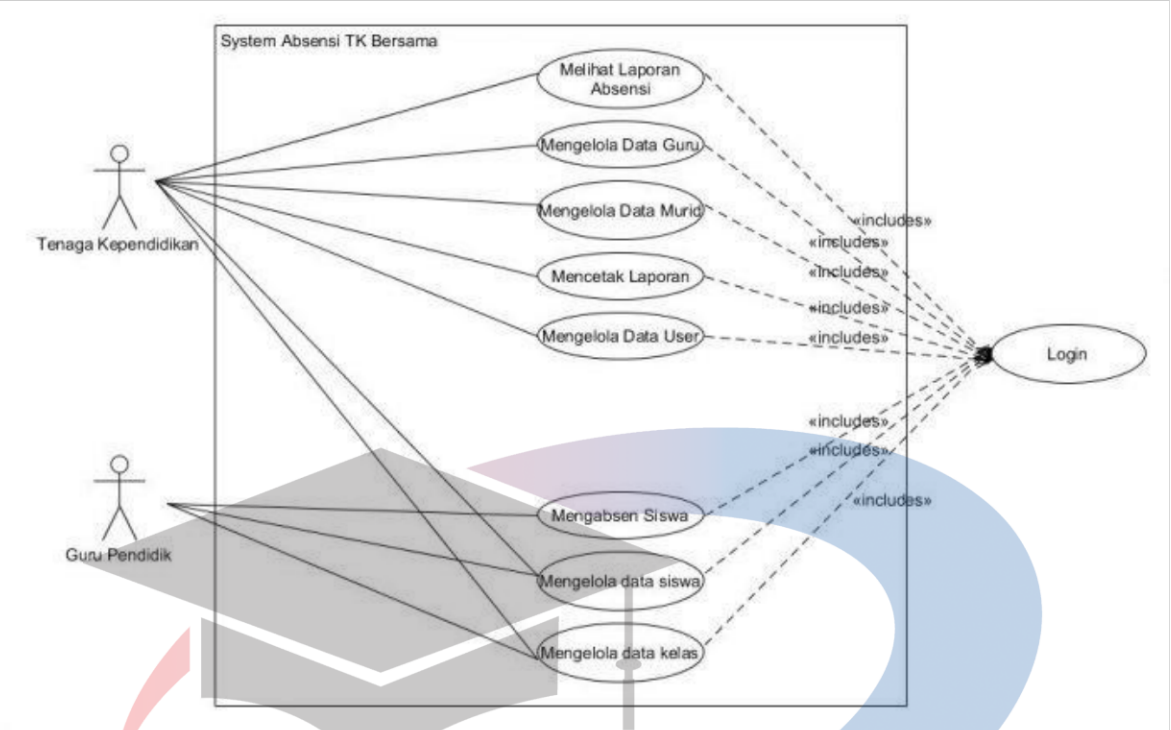
Use case selalu menggambarkan tiga hal: aktor yang memulai sebuah peristiwa, peristiwa yang memicu *use case* , dan *use case* yang melakukan tindakan yang dipicu oleh peristiwa tersebut. Dalam *use case* , seorang aktor menggunakan sistem memulai peristiwa yang memulai serangkaian interaksi terkait dalam sistem [7].

Tabel 2. 1 simbol use case diagram

NO	BENTUK SIMBOL	NAMA SIMBOL	FUNGSI SIMBOL
1.		*Actor*	Menspesifikasikan impunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2.		*Depedency*	Menyatakan hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung pada elemen yangtidak mandiri
3.		*Generalization*	Menunjukan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
4.		*Include*	Menunjukan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
5.		*Extend*	Menunjukan bahwa suatu *use case* merupakan fungsionalitas dari *use case* lainnya jika suatu kondisi terpenuhi
6.		*Association*	Menyatakan abstraksi dari penghubung antara aktor dengan *use case*

7.		*Usecase*	Menyatakan abstraksi dan interaksi antara sistem dan actor
8.		*Collaboration*	Menunjukan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya.
9.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

Deskripsi sebuah *use case* dapat melibatkan banyak In*form*asi, dan diperlukan cara sistematis untuk merekam In*form*asi ini. Namun, UML tidak menentukan cara standar untuk menyajikan deskripsi *use case* . Bagian dari alasan ini adalah bahwa *use case* dimaksudkan untuk digunakan secara in*form*al, sebagai bantuan dalam berkomunikasi dengan pengguna - pengguna masa depan suatu sistem, dan penting bahwa pengembang harus memiliki kebebasan untuk membahas *use case* dengan cara apa pun yang tampak membantu dan dapat dimengerti oleh pengguna [7].



Gambar 2.3 Contoh use case diagram

2.3.2 Sequence Diagram

Sequence Diagram dapat menggambarkan serangkaian interaksi antara kelas atau instance objek dalam waktu. Diagram urutan sering digunakan untuk menggambarkan proses yang dijelaskan dalam skenario penggunaan kasus. Dalam praktiknya, diagram urutan berasal dari analisis penggunaan kasus dan digunakan dalam desain sistem untuk mendapatkan interaksi, relasi, dan metode objek dalam sistem [7].

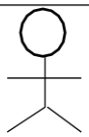
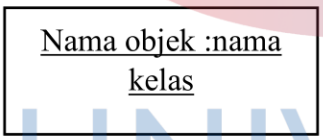
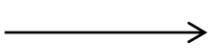
Diagram urutan digunakan untuk menunjukkan pola keseluruhan aktivitas atau interaksi dalam suatu penggunaan kasus. Setiap skenario penggunaan kasus dapat menciptakan satu diagram urutan, meskipun diagram urutan tidak selalu dibuat untuk skenario yang minor [7].

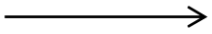
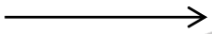
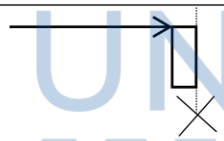
Sequence Diagram dapat berupa diagram urutan generik yang menunjukkan semua skenario mungkin untuk sebuah kasus penggunaan, tetapi biasanya setiap analis mengembangkan seperangkat diagram urutan *instance*, masing-masing menggambarkan sebuah skenario tunggal dalam kasus penggunaan. Jika Anda tertarik untuk memahami alur kontrol dari sebuah skenario dari waktu ke waktu, Anda harus menggunakan diagram urutan untuk menggambarkan In*form*asi ini. Diagram tersebut digunakan sepanjang tahap analisis dan

desain. Namun, diagram desain sangat spesifik pada implementasi, sering kali termasuk objek *database* atau komponen antarmuka pengguna spesifik sebagai objek [5].

Sequence Diagram berisi simbol aktor dan kasus penggunaan, serta garis penghubung. Aktor mirip dengan entitas eksternal; mereka ada di luar sistem. Istilah aktor merujuk pada peran tertentu pengguna sistem.

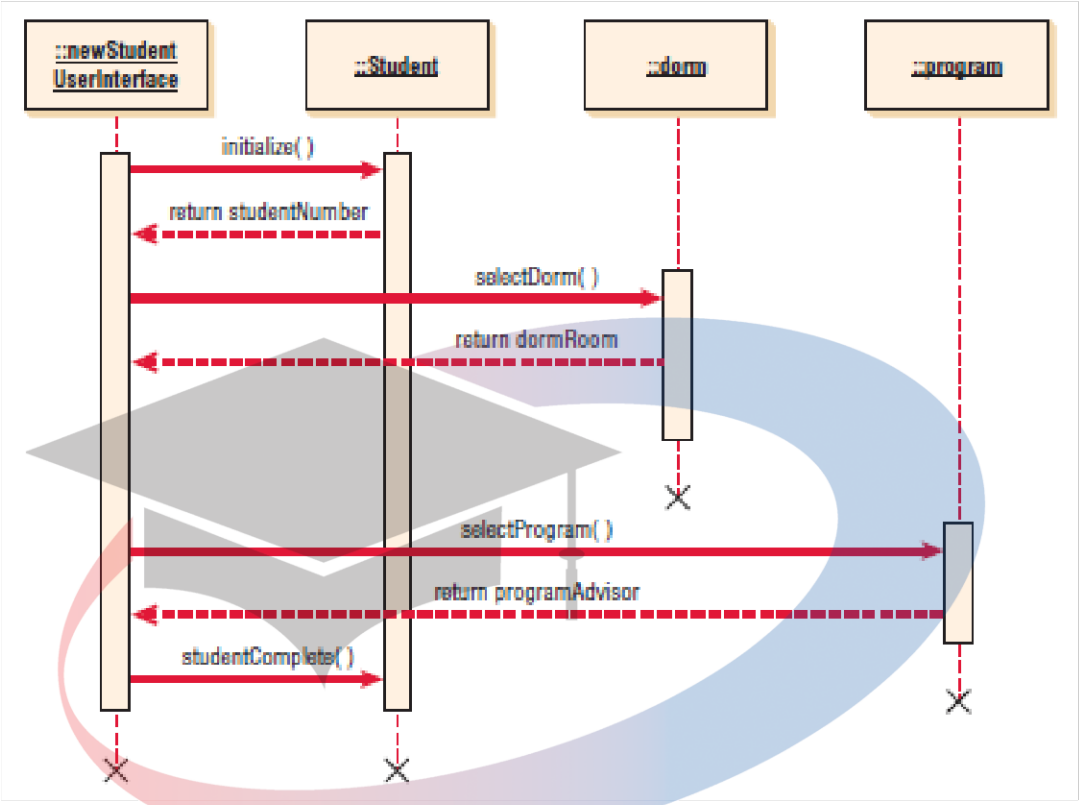
Tabel 2. 2 simbol sequence diagram

NO	BENTUK SIMBOL	NAMA SIMBOL	FUNGSI SIMBOL
1		Actor	Orang, proses, atau system lain yang berinteraksi dengan system In*form*asi yang akan dibuat itu sendiri.
2		Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek
3		Objek	Menyatakan objek yangberinteraksi pesan
4		Waktu aktif	semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
5	<<create>> 	Pesan tipe *create*	Menyatakan suatu objek membuat objek yang lain, arah panah

			mengarah pada objek yang dibuat.
6	1 : nama_metode 	Pesan tipe *call*	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
7	1 : masukan 	Pesan tipe *send*	Menyatakan abstraksi dan interaksi antara sistem dan actor
8	1 : keluaran 	Pesan tipe *return*	Menunjukan elemen lain yang bekerja sama untuk menyediakanprilaku yang lebih besar dari jumlah dan elemen-elemennya.
9		Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain,arah panah mengarah pada objek yang di akhiri.

Sebagai contoh, aktor dapat menjadi karyawan, tetapi juga dapat menjadi pelanggan di toko perusahaan. Meskipun di dunia nyata orang yang sama, aktor direpresentasikan sebagai dua simbol yang berbeda pada diagram kasus penggunaan karena orang tersebut berinteraksi dengan sistem dalam peran yang berbeda. Aktor ada di luar sistem dan berinteraksi dengan sistem dengan cara tertentu. Aktor dapat berupa manusia, sistem lain, atau perangkat seperti

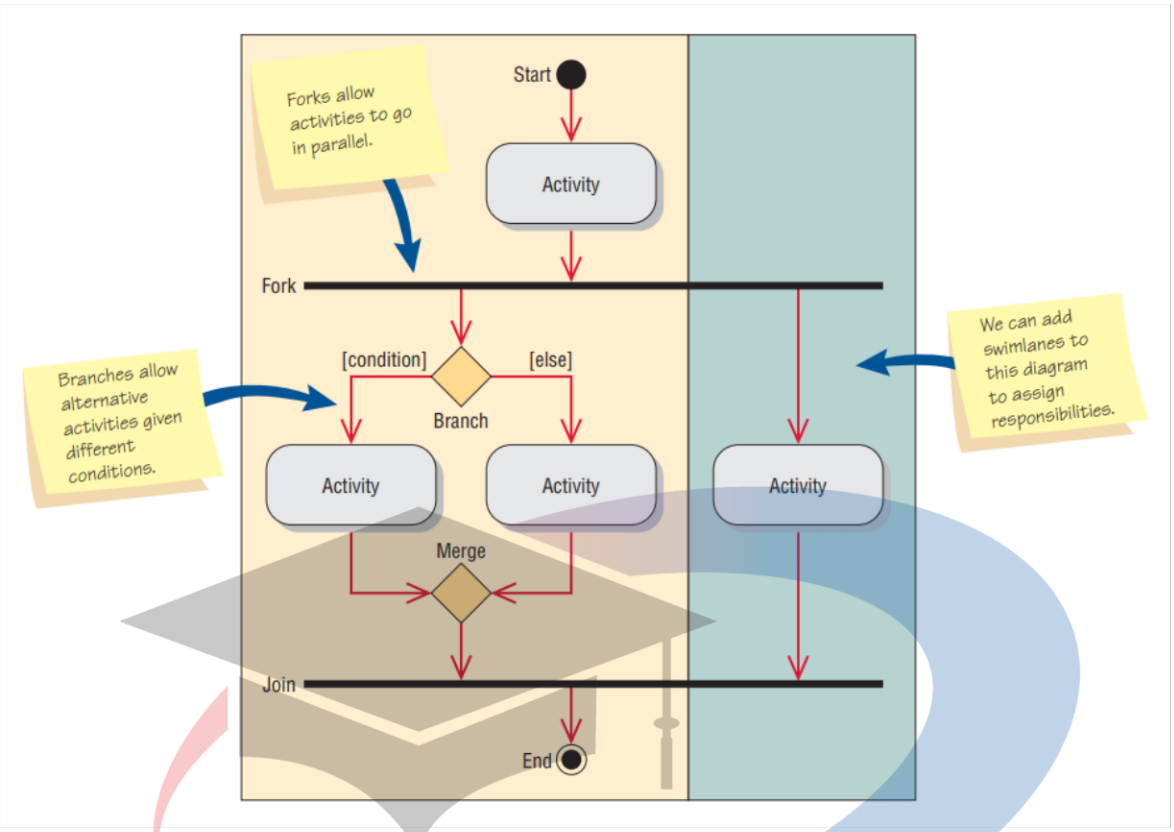
keyboard atau koneksi Web. Aktor dapat memulai sebuah kasus penggunaan. Seorang aktor dapat berinteraksi dengan satu atau lebih kasus penggunaan, dan sebuah kasus penggunaan dapat melibatkan satu atau lebih aktor [8].



Gambar 2.4 Contoh Sequence diagram

2.3.3 Activity Diagram

Diagram *Activity*/aktivitas menunjukkan urutan aktivitas dalam suatu proses, termasuk aktivitas berurutan dan paralel, serta keputusan yang dibuat. Diagram aktivitas biasanya dibuat untuk satu kasus penggunaan dan dapat menunjukkan kemungkinan skenario yang berbeda [7].



Gambar 2.5 Penjelasan Activity diagram

Simbol-simbol pada diagram aktivitas berupa persegi panjang dengan ujung membulat mewakili aktivitas, baik aktivitas manual, seperti menandatangani dokumen resmi, atau aktivitas otomatis, seperti metode atau program. Panah mewakili suatu peristiwa. Peristiwa mewakili hal-hal yang terjadi pada waktu tertentu dan tempat [7].

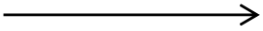
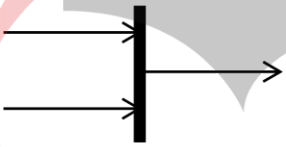
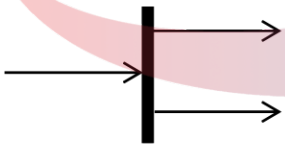
Berlian mewakili keputusan (juga disebut cabang) atau penggabungan. Keputusan memiliki satu panah masuk ke berlian dan beberapa keluar. Kondisi penjaga, yang menunjukkan nilai kondisi, dapat disertakan. Penggabungan memperlihatkan beberapa peristiwa yang digabungkan untuk membentuk satu peristiwa [7].

Persegi panjang yang panjang dan datar mewakili bilah sinkronisasi. Ini digunakan untuk menunjukkan aktivitas paralel dan mungkin memiliki satu peristiwa masuk ke bilah sinkronisasi dan beberapa peristiwa keluar darinya, yang disebut garpu. Sinkronisasi di mana beberapa acara digabungkan menjadi satu acara disebut gabungan [7].

Simbol-simbol yang terdapat pada *Activity* Diagram:

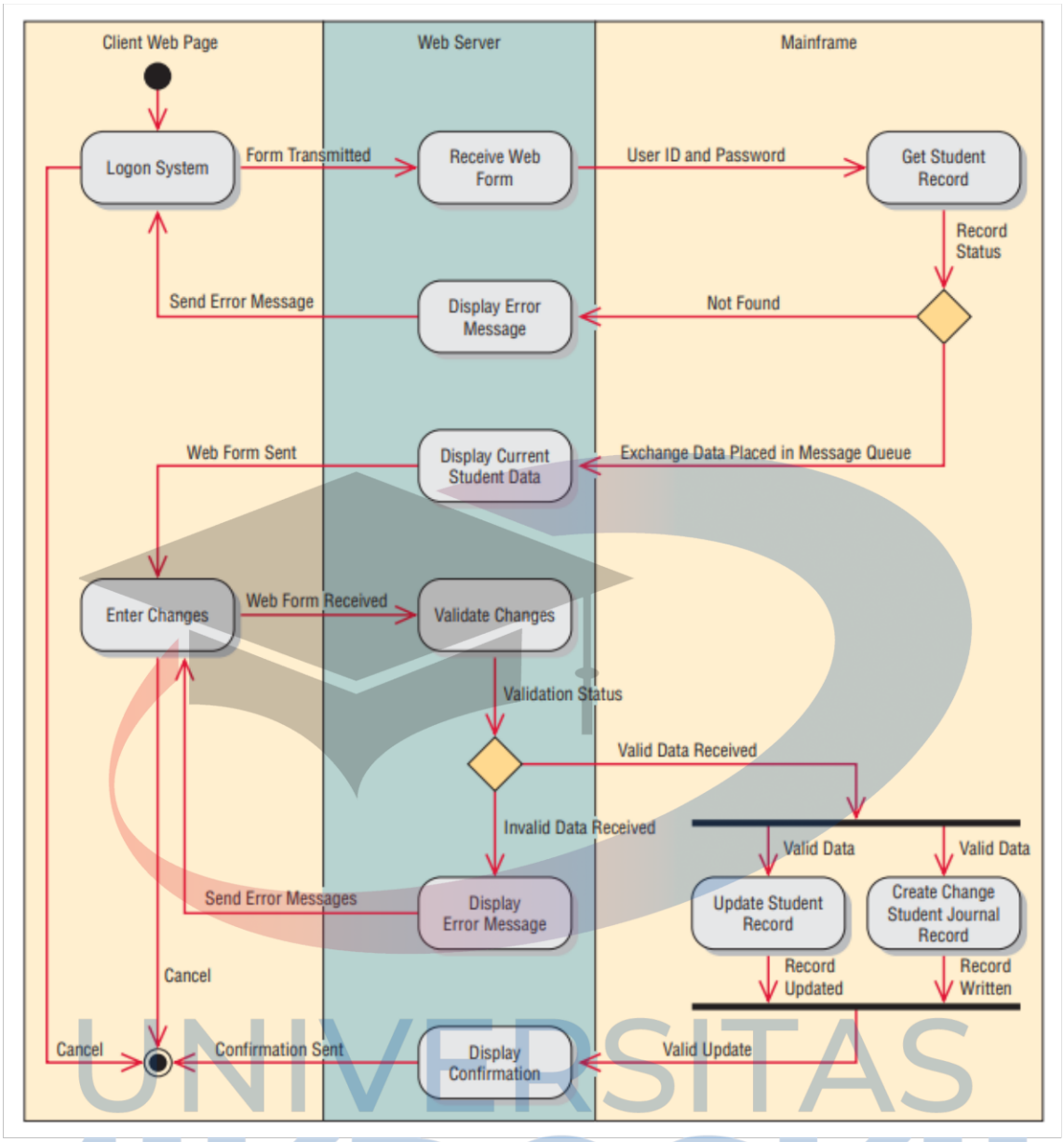
Tabel 2. 3 simbol activity diagram

NO	BENTUK SIMBOL	NAMA SIMBOL	FUNGSI SIMBOL

1.		*Activity*	Menyatakan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2.		*Control Flow*	Menunjukan Urutan Eksekusi.
3.		*Object Flow*	Menunjukan aliran objek dari sebuah action atau *activity* ke action.
4.		*Start Point*	Menyatakan bahwa sebuah objek dibentuk atau diawali.
5.		*End Point*	Menyatakan bahwa sebuah objek dibentuk atau diakhiri.
6.		*Join/Penggabungan*	Menyatakan untuk menggabungkan kembali *activity* atau action yang parallel.
7.		*Fork*	Menyatakan untuk memecah behavior menjadi *activity* atau action yang parallel.
8.		*Decision*	Menunjukan penggambaran suatu keputusan/tindakan yang harus di ambil pada kondisitertentu.

Ada dua simbol yang menunjukkan awal dan akhir diagram. Keadaan awal ditampilkan sebagai lingkaran yang terisi. Keadaan akhir ditampilkan sebagai lingkaran hitam yang dikelilingi oleh lingkaran putih [7].

Persegi panjang yang mengelilingi simbol lain, yang disebut swimlanes, menunjukkan partisi dan digunakan untuk menunjukkan aktivitas mana yang dilakukan pada plat*form* mana, seperti browser, server, atau komputer *mainframe*, atau untuk menunjukkan aktivitas yang dilakukan oleh kelompok pengguna yang berbeda. Swimlanes adalah zona yang dapat menggambarkan logika sekaligus tanggung jawab sebuah kelas [7].



Gambar 2.6 Contoh Activity Diagram

Anda dapat melihat contoh diatas, yang mengilustrasikan diagram aktivitas untuk *use case Change Student Information*. Ini dimulai dengan siswa masuk ke sistem dengan mengisi *form*ulir Web dan mengklik tombol Kirim. *Form*ulir dikirimkan ke server web, yang kemudian meneruskan data ke komputer *mainframe*. *Mainframe* mengakses *database* SISWA dan mengirimkan pesan "Tidak Ditemukan" atau data siswa yang dipilih ke server web [7].

Berlian di bawah status Get Student Record menunjukkan keputusan ini. Jika record siswa belum ditemukan, web server menampilkan pesan kesalahan pada halaman web. Jika catatan siswa telah ditemukan, server web mem*form*at halaman web baru yang berisi data siswa saat ini dalam *form*ulir Web. Siswa dapat membatalkan perubahan baik dari Sistem Masuk atau status Masuki Perubahan, dan aktivitas dihentikan [7].

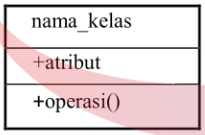
Jika siswa memasukkan perubahan pada *form*ulir Web dan mengklik tombol Kirim, data perubahan dikirimkan ke server dan program mulai berjalan yang memvalidasi perubahan. Jika ada kesalahan, pesan kesalahan dikirim ke halaman web. Jika data sudah valid, record siswa diperbarui dan dibuatkan Change Student Journal Record. Setelah pembaruan yang valid, halaman web konfirmasi dikirimkan ke browser, dan aktivitas dihentikan [7].

2.3.4 *Class* Diagram

Metodologi berorientasi objek bekerja untuk menemukan kelas, atribut, metode, dan hubungan antar kelas. Karena pemrograman terjadi pada level kelas, mendefinisikan kelas adalah salah satu tugas analisis berorientasi objek yang paling penting. Diagram kelas menunjukkan fitur statis dari sistem dan tidak mewakili pemrosesan tertentu. Diagram kelas juga menunjukkan sifat hubungan antar kelas [7].

Simbol-simbol yang terdapat pada *Class* Diagram :

Tabel 2. 4 simbol class diagram

NO	BENTUK SIMBOL	NAMA SIMBOL	FUNGSI SIMBOL
1.		Kelas	Kelas pada struktur sistem
2.		Antarmuka/*interface*	Sama dengan konsep interface dalam pemrograman berorientasiobjek
3.		Asosiasi/*association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
4.		Asosiasi berarah/*directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
5.		Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)

6.		Kebergantungan/*depedency*	Relasi antar kelas dengan kebergantungan antar kelas
7.		Agregasi/*aggregation*	Relasi antar kelas dengan makna semua bagian (whole-part)

Kelas diwakili oleh persegi panjang pada diagram kelas. Dalam *form*at yang paling sederhana, persegi panjang mungkin hanya menyertakan nama kelas, tetapi juga dapat menyertakan atribut dan metode. Atribut adalah apa yang diketahui kelas tentang karakteristik objek, dan metode (juga disebut operasi) adalah apa yang diketahui kelas tentang cara melakukan sesuatu. Metode adalah bagian kecil dari kode yang bekerja dengan atribut [7].

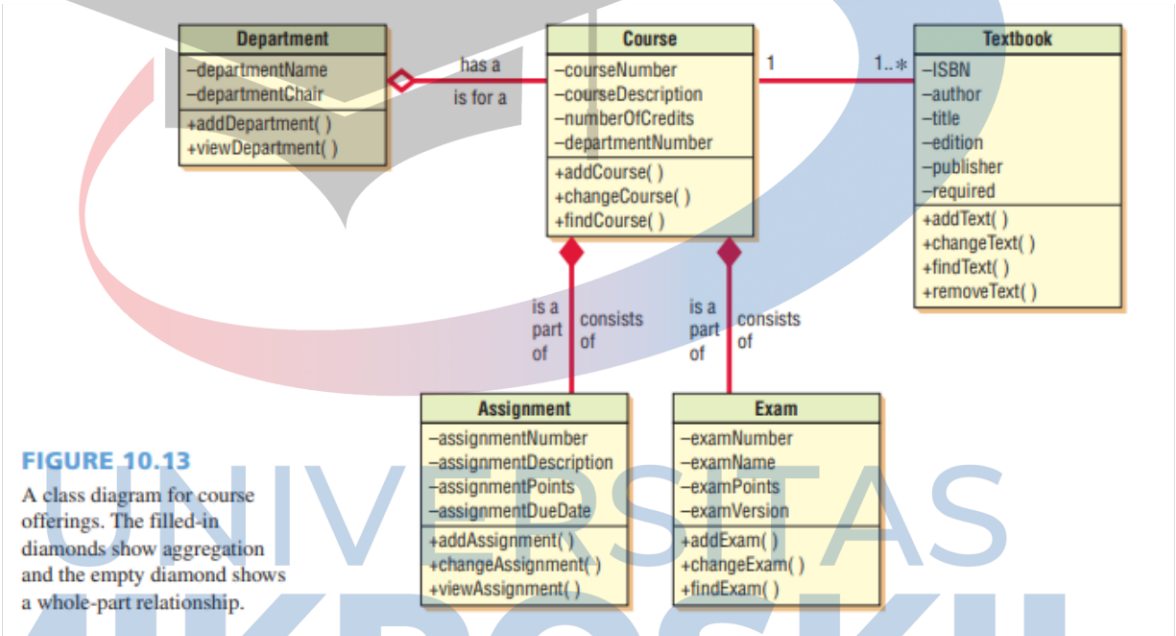


FIGURE 10.13
A class diagram for course offerings. The filled-in diamonds show aggregation and the empty diamond shows a whole-part relationship.

Gambar 2.7 Diagram kelas untuk kursus

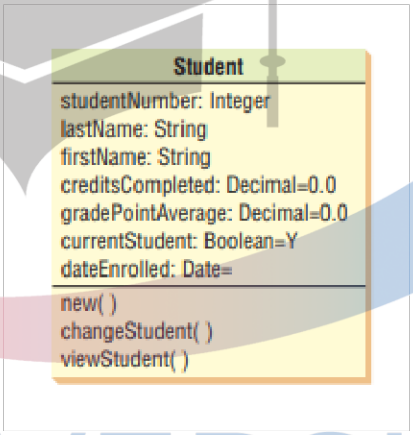
Gambar di atas mengilustrasikan diagram kelas untuk penawaran kursus. Perhatikan bahwa nama dipusatkan di bagian atas kelas, biasanya dengan huruf tebal. Area tepat di bawah nama menampilkan atribut, dan bagian bawah mencantumkan metode. Diagram kelas menunjukkan penyimpanan data [7].

persyaratan serta persyaratan pemrosesan. Nanti di bab ini kita akan membahas arti dari simbol berlian yang ditunjukkan pada gambar ini. Atribut (atau properti) biasanya ditetapkan sebagai pribadi, atau hanya tersedia di obyek. Ini diwakili pada diagram kelas dengan tanda minus di depan nama atribut Atribut juga dapat dilindungi, ditunjukkan dengan simbol pound (#). Atribut ini disembunyikan dari semua kelas kecuali subkelas langsung. Dalam keadaan

yang jarang terjadi, atribut bersifat publik, artinya dapat dilihat oleh objek lain di luar kelasnya. Menjadikan atribut pribadi berarti atribut hanya tersedia untuk objek luar melalui metode kelas, teknik yang disebut enkapsulasi, atau penyembunyian In*form*asi [7].

Diagram kelas mungkin hanya menampilkan nama kelas; atau nama kelas dan atribut; atau nama kelas, atribut, dan metode. Menampilkan hanya nama kelas berguna ketika diagramnya rumit dan mencakup banyak kelas. Jika diagram lebih sederhana, atribut dan metode dapat disertakan. Saat atribut disertakan, ada tiga cara untuk menampilkan In*form*asi atribut. Yang paling sederhana adalah dengan memasukkan hanya nama atribut, yang membutuhkan ruang paling sedikit [7].

Jenis data (seperti *string, double, integer*, atau tanggal) dapat disertakan pada diagram kelas. Deskripsi paling lengkap akan menyertakan tanda sama dengan (=) setelah jenis data, diikuti dengan nilai awal atribut. Gambar di bawah ini mengilustrasikan atribut kelas [7].



Gambar 2.8 Gambar Siswa yang di perluas

Jika atribut harus mengambil salah satu dari jumlah nilai yang terbatas, seperti tipe siswa dengan nilai F untuk waktu penuh, P untuk waktu paruh waktu, dan N untuk nonmatrikulasi, ini dapat dimasukkan dalam tanda kurung kurawal yang dipisahkan dengan koma sebagai ditampilkan di sini studentType:char{F.P,N) [7].

Penyembunyian In*form*asi berarti bahwa metode objek harus tersedia untuk kelas lain, sehingga metode seringkali bersifat publik, artinya metode tersebut dapat dipanggil dari kelas lain. Pada diagram kelas. pesan publik (dan semua atribut publik) ditampilkan dengan tanda plus (+) di depannya. Metode juga memiliki tanda kurung setelahnya, yang menunjukkan bahwa data dapat diteruskan sebagai parameter bersama dengan pesan. Parameter pesan, serta tipe data, dapat disertakan pada diagram kelas [7].

2.4 *Database* (Basis Data)

Elemen basis data pada sistem In*form*asi berfungsi sebagai media untuk penyimpanan data dan In*form*asi yang dimiliki oleh sistem In*form*asi bersangkutan. Setiap aplikasi dan sistem yang memiliki data di dalamnya (dengan disertai proses manipulasi data berupa *insert*, *delete*, *edit*/*update*), pasti memiliki sebuah basis data [4].

Umumnya, sebuah basis data memiliki satu atau beberapa buah tabel. Setiap tabel memiliki *field* masing-masing. Ke dalam tabel dan *field* inilah data disimpan oleh pengguna melalui tatap muka aplikasi yang disediakan atau langsung melalui perintah di terminal (*command line*) [4].

Basis data bukan hanya kumpulan berkas belaka. Sebaliknya, basis data adalah sumber data pusat yang dimaksudkan untuk dibagikan oleh banyak pengguna untuk berbagai aplikasi. Inti dari basis data adalah sistem manajemen basis data (DBMS), yang memungkinkan pembuatan, modifikasi, dan pembaruan basis data; pengambilan data; dan penghasilan laporan dan tampilan. Orang yang memastikan basis data mencapai tujuannya disebut administrator basis data [7].

Tujuan efektivitas dari basis data meliputi hal-hal berikut: [7]

1. Memastikan bahwa data dapat dibagi antara pengguna untuk berbagai aplikasi.
2. Memelihara data yang akurat dan konsisten.
3. Memastikan bahwa semua data yang diperlukan untuk aplikasi saat ini dan di masa depan akan tersedia dengan mudah.
4. Memungkinkan basis data berkembang seiring dengan kebutuhan pengguna.
5. Memungkinkan pengguna untuk membuat pandangan pribadi mereka terhadap data tanpa perlu khawatir tentang cara penyimpanan data secara fisik.