

ANALISA DAN PERANCANGAN SISTEM

Penyusun :

Irpan Kusyadi

Maulana Ardhiansyah

Hidayatullah Al Islami



Jl. Surya Kencana No. 1 Pamulang
Gd. A, Ruang 211 Universitas Pamulang
Tangerang Selatan – Banten

ANALISA DAN PERANCANGAN SISTEM

Penulis :

Irpan Kusyadi

Maulana Ardhiansyah

Hidayatullah Al Islami

ISBN : 978-623-6352-25-0

Editor :

Rahmawati

Desain Sampul:

Tri Anggoro Seto

Tata Letak:

Ramdani Putra

Penerbit:

Unpam Press

Redaksi:

Jl. Surya Kencana No. 1

R. 212, Gd. A Universitas Pamulang Pamulang | Tangerang Selatan | Banten

Tlp/Fax: 021. 741 2566 – 7470 9855 Ext: 1073

Email: unpampress@unpam.ac.id

Cetakan pertama, 9 Agustus 2021

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa izin penerbit.

DATA PUBLIKASI UNPAM PRESS

I Pusat Kajian Pembelajaran & E-learning Universitas Pamulang

Gedung A. R.212 Kampus 1 Universitas Pamulang

Jalan Surya Kencana No.1, Pamulang Barat, Tangerang Selatan, Banten.

Website : www.unpam.ac.id **I email :** unpampress@unpam.ac.id

Analisa dan Perancangan Sistem / Irpan Kusyadi, Maulana Ardhiansyah, dan Hidayatullah Al Islami -1STed

ISBN. 978-623-6352-25-0

1. Analisa dan Perancangan Sistem I. Irpan Kusyadi II. Maulana Ardhiansyah III. Hidayatullah Al Islami

M162-09082021-01

Ketua Unpam Press: Pranoto

Koordinator Editorial: Aden, Ali Madinsyah

Koordinator Hak Cipta: Susanto

Koordinator Produksi: Dameis Surya Anggara

Koordinator Publikasi dan Dokumentasi: Kusworo

Desain Cover: Putut Said Permana

Cetakan pertama, 9 Agustus 2021

Hak cipta dilindungi undang-undang. Dilarang menggandakan dan memperbanyak sebagian atau seluruh buku ini dalam bentuk dan dengan cara apapun tanpa ijin penerbit.

MATA KULIAH
ANALISA DAN PERANCANGAN SISTEM

IDENTITAS MATA KULIAH

Program Studi	: Analisa dan Perancangan Sistem/ TPL0282
Sks	: 2 Sks
Prasyarat	: -
Semester	: 5
Deskripsi Mata Kuliah	: Mata kuliah Analisa dan perancangan Sistem merupakan mata kuliah wajib Program Studi Teknik Informatika S-1 yang membahas pengertian sistem, pengembangan sistem, analisis sistem, data flow diagram, flowchart, metodologi berorientasi obyek, diagram-diagram UML dan melakukan perancangan sistem dengan model UML
Capaian Pembelajaran	: Setelah menyelesaikan mata kuliah ini, mahasiswa mampu merancang, mendesain dan menganalisis sistem dengan Metodologi Berorientasi Obyek serta diharapkan dapat mendisain sistem dengan diagram-diagram UML sesuai dengan kebutuhan sistem.
Penyusun	: 1. Irpan Kusyadi 2. Maulana Ardhiansyah 3. Hidayatullah Al Islami

Ketua Program Studi
Teknik Informatika S-1

Ketua Tim Penyusun

Achmad Udin Zailani, S.Kom., M.Kom.

Irpan Kusyadi, S. Kom., M. Kom.

NIDN : 0429058303

NIDN. 0411109001

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan hidayat-Nya sehingga modul bahan ajar Analisa dan Perancangan Sistem dapat terselesaikan.

Teknologi berkembang semakin pesat, pembangunan sistem tidak terlepas dari berkembangnya teknologi. Sebuah sistem yang baik akan berdampak terhadap penggunaan teknologi menjadi lebih baik, maka dalam membangun sistem yang baik diperlukan analisa dan perancangan sistem yang baik pula.

Analisa sistem merupakan teori sistem umum sebagai landasan konseptual yang bertujuan untuk memperbaiki berbagai fungsi didalam sistem yang sedang berjalan agar menjadi lebih efisien, mengubah sasaran sistem yang sedang berjalan, merancang output yang sedang digunakan, untuk mencapai tujuan yang sama dengan seperangkat input yang lain bisa berupa lebih sederhana dan atau lebih menarik dari sebelumnya.

Mata kuliah **Analisa dan Perancangan Sistem** mempelajari tentang materi pengertian sistem, pengembangan sistem, analisis sistem, data flow diagram, flowchart, metodologi berorientasi obyek, diagram-diagram UML dan melakukan perancangan sistem dengan model UML. Bahan Ajar ini disusun untuk memudahkan mahasiswa dalam mempelajari dan memahami serta mengaplikasikan materi mata kuliah Analisa dan Perancangan Sistem.

Akhirnya penulis sampaikan terimakasih atas semua bantuan dan dukungan dari semua pihak yang tidak disebutkan satu persatu dalam proses penulisan. Semoga bahan ajar ini dapat bermanfaat bagi kita semua.

Tangerang Selatan, 20 Maret 2021

Tim Penyusun

DAFTAR ISI

COVER.....	I
EDITORIAL	II
DATA PUBLIKASI UNPAM PRESS.....	III
IDENTITAS MATA KULIAH	IV
KATA PENGANTAR	V
DAFTAR ISI.....	VI
DAFTAR GAMBAR	XII
PERTEMUAN 1	1
SISTEM.....	1
A. TUJUAN PEMBELAJARAN	1
B. URAIAN MATERI	1
1. <i>Pengertian Sistem</i>	1
2. <i>Dasar Pemikiran Definisi Sistem</i>	1
3. <i>Karakteristik Sistem</i>	3
4. <i>Klasifikasi Sistem</i>	7
5. <i>Pendekatan dan Metodologi Pengembangan Sistem</i>	9
C. SOAL LATIHAN/ TUGAS	24
D. REFERENSI	24
PERTEMUAN 2	26
SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)	26
A. TUJUAN PEMBELAJARAN	26
B. URAIAN MATERI	26
1. <i>Pengertian SDLC</i>	26
2. <i>Sejarah Pengembangan SDLC</i>	27
3. <i>Tahapan SDLC</i>	32
4. <i>Implementation/ Implementasi Sistem</i>	35
5. <i>Alat Pengembangan</i>	37
C. SOAL LATIHAN/TUGAS	40

D. REFERENSI	40
PERTEMUAN 3	42
PERENCANAAN SISTEM	42
A. TUJUAN PEMBELAJARAN	42
B. URAIAN MATERI	42
1. <i>Pengertian Perencanaan</i>	42
2. <i>Perlunya Perencanaan</i>	43
3. <i>Proses Perencanaan Sistem</i>	44
C. SOAL LATIHAN/TUGAS	56
D. REFERENSI	56
PERTEMUAN 4	58
ANALISA SISTEM	58
A. TUJUAN PEMBELAJARAN	58
B. URAIAN MATERI	58
1. <i>Pengertian Analisa Sistem</i>	58
2. <i>Proses Analisa Sistem</i>	60
3. <i>Jenis Kebutuhan Sistem</i>	64
4. <i>Teknik Penghimpunan Data</i>	66
C. SOAL LATIHAN/TUGAS	73
D. REFERENSI	73
PERTEMUAN 5	75
MODELING PADA DFD	75
A. TUJUAN PEMBELAJARAN	75
B. URAIAN MATERI	75
1. <i>Pengertian Proses Modelling</i>	75
2. <i>Data Flow Diagram</i>	77
3. <i>Kegunaan DFD</i>	77
4. <i>Syarat-syarat Pembuatan Data Flow Diagram</i>	77
5. <i>Penggambaran DFD</i>	78
6. <i>Komponen Data Flow Diagram</i>	80
7. <i>Bentuk Data Flow Diagram</i>	84

C. SOAL LATIHAN/TUGAS	87
D. REFERENSI	87
PERTEMUAN 6	89
FLOWCHART	89
A. TUJUAN PEMBELAJARAN	89
B. URAIAN MATERI	89
1. <i>Pengertian Flowchart</i>	89
2. <i>Pedoman-Pedoman untuk Membuat Flowchart</i>	89
3. <i>Simbol-Simbol Flowchart</i>	91
4. <i>Jenis-Jenis Flowchart</i>	96
C. SOAL LATIHAN/TUGAS	103
D. REFERENSI	104
PERTEMUAN 7	106
ORIENTASI OBJEK	106
A. TUJUAN PEMBELAJARAN	106
B. URAIAN MATERI	106
1. <i>Konsep Perancangan Berorientasi Objek</i>	106
2. <i>Konteks Sistem dan Model Penggunaan</i>	108
3. <i>Diagram Objek</i>	110
4. <i>Perancangan Arsitektural</i>	113
C. SOAL LATIHAN/TUGAS	115
D. REFERESI	115
PERTEMUAN 8	117
ORIENTASI OBJEK (LANJUTAN)	117
A. TUJUAN PEMBELAJARAN	117
B. URAIAN MATERI	117
1. <i>Identifikasi Objek</i>	117
2. <i>Model Desain</i>	118
3. <i>Spesifikasi Interface Objek</i>	121
4. <i>Proses Desain Interface</i>	123

C. SOAL LATIHAN/TUGAS	125
D. REFERENSI	125
PERTEMUAN 9	127
Pengenalan UML	127
A. TUJUAN PEMBELAJARAN	127
B. URAIAN MATERI	127
1. <i>Pengenalan UML</i>	127
2. <i>Pengertian UML</i>	128
3. <i>Bagian - bagian dari UML</i>	129
4. <i>Gambaran dari UML</i>	135
5. <i>Area Penggunaan UML</i>	136
6. <i>Komponen – komponen yang terlibat dalam Use Case Diagram</i>	137
7. <i>Class Diagram</i>	138
8. <i>Interaction Diagram</i>	140
9. <i>State Transition Diagram dan Activity Diagram</i>	141
10. <i>Activity Diagram</i>	144
11. <i>Kafinment</i>	145
12. <i>Sejarah Singkat UML</i>	145
C. SOAL LATIHAN/TUGAS	146
D. REFERENSI	147
PERTEMUAN 10	149
BAGIAN DAN LANGKAH PEMBUATAN UML	149
A. TUJUAN PEMBELAJARAN	149
B. URAIAN MATERI	149
1. <i>Use Case Diagram</i>	149
2. <i>Activity Diagram</i>	150
3. <i>Package Diagram</i>	151
4. <i>State Diagram</i>	152
5. <i>Sequence Diagram</i>	153
6. <i>Class Diagram</i>	154
7. <i>Object Diagram</i>	155
8. <i>Collaboration Diagram</i>	156

9. <i>Component Diagram</i>	156
10. <i>Deployment Diagram</i>	157
11. <i>Interaction overview diagram</i>	157
12. <i>Conceptual Diagram</i>	158
13. <i>Timing diagram</i>	159
14. <i>Langkah – langkah pembuatan UML</i>	160
C. SOAL LATIHAN/TUGAS	162
D. REFERENSI	162
PERTEMUAN 11	164
IMPLEMENTASI DIAGRAM UML	164
A. TUJUAN PMBELAJARAN.....	164
B. URAIAN MATERI	164
1. <i>Use Case Diagram</i>	164
2. <i>Class Diagram</i>	170
3. <i>Object Diagram</i>	174
C. SOAL LATIHAN/TUGAS	177
D. REFERENSI	177
PERTEMUAN 12	179
IMPLEMENTASI DIAGRAM UML PADA STATECHART DIAGRAM DAN ACTIVITY DIAGRAM	179
A. TUJUAN PEMBELAJARAN	179
B. URAIAN MATERI	179
1. <i>Statechart Diagram</i>	179
2. <i>Activity Diagram</i>	188
C. SOAL LATIHAN/TUGAS	193
D. REFERENSI	193
PERTEMUAN 13	195
IMPLEMENTASI PERANCANGAN UML LANJUTAN SEQUENCE DIAGRAM, COLLABORATION DIAGRAM, COMPONENT DIAGRAM, DEPLOYMENT DIAGRAM	195
A. TUJUAN PEMBELAJARAN	195

B. URAIAN MATERI	195
1. <i>Sequence Diagram</i>	195
2. <i>Collaboration Diagram</i>	198
3. <i>Component Diagram</i>	200
C. SOAL LATIHAN/TUGAS	207
D. REFERENSI	207
PERTEMUAN 14	209
STUDI KASUS PERANCANGAN DIAGRAM UML PEMBUATAN SUATU SISTEM	209
A. TUJUAN PEMBELAJARAN	209
B. URAIAN MATERI	209
1. <i>Perancangan Studi kasus Dilapangan</i>	209
2. <i>Perancangan Use Case Diagram</i>	209
3. <i>Perancangan Class Diagram</i>	210
4. <i>Activity Diagram</i>	214
5. <i>Sequence diagram</i>	218
C. SOAL LATIHAN/TUGAS	223
D. REFERENSI	223

DAFTAR GAMBAR

Gambar 1 Metodologi Pengembangan Waterfall	11
Gambar 2 Metodologi Pengembangan Paralel	12
Gambar 3. Contoh state mesin menyalakan lampu bohlam	182
Gambar 4. Contoh penggambaran nilai atribut.	184
Gambar 5. Contoh state ketika memasukan suatu password	184
Gambar 6. Gambar sebuah transisi pada Statechart.....	185
Gambar 7. Gambar event pada suatu state mesin.....	186
Gambar 8. Statechart diagram untuk melakukan login pada aplikasi.....	186
Gambar 9. Statechart diagram pada sistem pemesanan	187

PERTEMUAN 1

SISTEM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang apa itu pengertian sistem, karakteristik sistem, klasifikasi sistem, juga pendekatan dan metodologi pengembangan sistem. Setelah pertemuan ini, mahasiswa mampu mendeskripsikan klasifikasi dan metodologi pengembangan sistem.

B. URAIAN MATERI

1. Pengertian Sistem

Istilah "sistem" diambil dari bahasa Yunani: systēma, yang berarti "menempatkan bersama." Berbagai domain bisnis dan teknik memiliki definisi sistem. Teks ini mendefinisikan sistem sebagai:

"Sistem adalah sebuah set elemen yang dapat dioperasikan dan saling terintegrasi, masing-masing ditentukan dengan secara eksplisit dan kemampuan terbatas, bekerja secara sinergis untuk melakukan pemrosesan nilai tambah yang memungkinkan Pengguna untuk memenuhi kebutuhan berorientasi misi dalam lingkungan operasi yang ditentukan dengan hasil tertentu dan kemungkinan sukses".

Untuk membantu memahami dasar pemikiran definisi ini, mari periksa setiap bagian secara mendetail.

2. Dasar Pemikiran Definisi Sistem

Definisi di atas menangkap sejumlah poin diskusi utama tentang sistem. Mari periksa dasar untuk setiap frasa dalam definisi.

- a. Yang dimaksud dengan "himpunan terintegrasi" adalah bahwa sistem, menurut definisi, terdiri dari hierarki tingkat elemen fisik, entitas, atau komponen.

- b. Yang dimaksud dengan "elemen yang dapat dioperasikan", yang dimaksud adalah bahwa elemen dalam struktur sistem haruslah kompatibel satu sama lain dalam bentuk, fit, dan fungsi, misalnya. Elemen sistem termasuk peralatan (misalnya, perangkat keras dan perangkat lunak, personel, fasilitas, kendala operasi, dukungan), pemeliharaan, persediaan, suku cadang, pelatihan, sumber daya, data prosedural, sistem eksternal, dan apa pun yang mendukung pencapaian misi.
- c. Dengan setiap elemen yang memiliki "kemampuan yang ditentukan dan dibatasi secara eksplisit", yang dimaksud adalah setiap elemen harus bekerja untuk mencapai beberapa tujuan tingkat yang lebih tinggi atau misi yang bertujuan. Sistem kontribusi elemen terhadap kinerja sistem secara keseluruhan harus ditentukan secara eksplisit. Ini membutuhkan kemampuan kinerja operasional dan fungsional untuk setiap elemen sistem dapat diidentifikasi dan secara eksplisit dibatasi ke tingkat kota tertentu yang memungkinkan elemen tersebut berada dianalisis, dirancang, dikembangkan, diuji, diverifikasi, dan divalidasi baik secara berdiri sendiri atau sebagai bagian dari sistem terintegrasi.
- d. Yang dimaksud dengan "bekerja secara sinergis" adalah tujuan mengintegrasikan himpunan elemen adalah memanfaatkan kapabilitas kapabilitas elemen individu untuk mencapai level yang lebih tinggi kemampuan yang tidak dapat dicapai sebagai elemen yang berdiri sendiri.
- e. Dengan "pemrosesan nilai tambah", maksudnya adalah faktor-faktor seperti biaya operasional, utilitas, kesesuaian, ketersediaan, dan efisiensi menuntut agar setiap operasi sistem dan tugas menambah nilai inputnya ketersediaan, dan menghasilkan keluaran yang berkontribusi pada pencapaian misi sistem secara keseluruhan hasil dan tujuan kinerja.
- f. Dengan "memungkinkan pengguna untuk secara terprediksi memenuhi kebutuhan berorientasi misi", maksudnya adalah setiap sistem memiliki tujuan (yaitu, alasan keberadaan) dan nilai bagi pengguna. Nilainya mungkin laba atas investasi (ROI) relatif untuk memenuhi kebutuhan operasional atau untuk memuaskan misi dan tujuan sistem.
- g. Dengan "dalam lingkungan operasi yang ditentukan", maksudnya adalah untuk ekonomi, hasil, dan untuk alasan kelangsungan hidup, setiap sistem harus memiliki operasi yang ditentukan yaitu, terikat lingkungan Hidup.
- h. Yang dimaksud dengan "dengan hasil yang ditentukan" adalah pemangku

kepentingan sistem (Pengguna, pemegang saham, pemilik, dll.) mengharapkan sistem memberikan hasil. Perilaku yang diamati, produk, oleh produk, atau layanan, misalnya, harus berorientasi pada hasil, dapat diukur, dapat diukur, dan bisa dibuktikan.

- i. Yang dimaksud dengan "dan kemungkinan sukses" adalah pencapaian hasil tertentu melibatkan tingkat ketidakpastian atau risiko. Jadi, derajat keberhasilan ditentukan oleh berbagai macam faktor kinerja seperti keandalan, ketergantungan, ketersediaan, pemeliharaan, keberlanjutan kemampuan, mematikan, dan bertahan hidup.

3. Karakteristik Sistem

Dalam mengkarakterisasi sistem, terutama untuk pemasaran atau analisis, ada empat tipe dasar karakteristik antara lain adalah:

a. Karakteristik Umum

Fitur tingkat tinggi dari suatu sistem adalah karakteristik umumnya. Karakter umum sering dinyatakan dalam brosur pemasaran di mana fitur-fitur utama ditekankan untuk menangkap klien atau minat pelanggan. Karakteristik umum sering kali memiliki beberapa kesamaan di beberapa contoh atau model sistem. Perhatikan contoh berikut:

- 1) Karakteristik Umum Mobil Tersedia dalam model dua pintu atau empat pintu; convertible atau sedan; AC nyaman; penanggulangan independen; jendela berwarna, kota 22mpg, jalan raya 30mpg.
- 2) Karakteristik Umum Pesawat Fanjet, 50 penumpang, jangkauan 2000 mil laut, kemampuan IFR.
- 3) Karakteristik Umum Perusahaan atau Organisasi 200 karyawan; staf dengan 20 PhD, 50 Master, dan 30 derajat BS; penjualan tahunan sebesar \$ 500 M per tahun.
- 4) Karakteristik Umum Jaringan Arsitektur server-klien, platform PC dan Unix, keamanan firewall, akses dial-up jarak jauh, tulang punggung Ethernet, struktur file jaringan (NFS).

b. Karakteristik Operasi atau Perilaku

Pada tingkat kerincian di bawah karakteristik umum, sistem memiliki karakteristik operasi itu menjelaskan fitur sistem yang terkait dengan

kegunaan, survivabilitas, dan kinerja untuk lingkungan operasi yang ditentukan. Perhatikan contoh berikut:

- 1) Karakteristik Operasi Mobil Kemampuan manuver, radius putar 18 kaki, 0 hingga 60 mph dalam 6 detik, dll.
- 2) Karakteristik Operasi Pesawat Aplikasi segala cuaca, kecepatan, dll.
- 3) Otorisasi Karakteristik Operasi Jaringan, waktu akses, latensi, dll.

c. Karakteristik fisik

Setiap sistem digambarkan oleh karakteristik fisik yang berhubungan dengan atribut nonfungsional tersebut sebagai atribut ukuran, berat, warna, kapasitas, dan antarmuka. Perhatikan contoh berikut:

- 1) Karakteristik Fisik Mobil 2000 lbs, berat trotoar volume kargo 14,0 cu ft, 43,1 inci (maks). ruang kaki depan, kapasitas bahan bakar 17,1 gals, mesin 240 tenaga kuda pada 6250 rpm, turbo, tersedia dalam 10 warna.
- 2) Karakteristik Fisik Perusahaan atau Organisasi Ruang kantor seluas 5.000 kaki persegi, 15 komputer jaringan, 100.000 kaki persegi gudang.
- 3) Karakteristik Fisik Jaringan tulang punggung Ethernet 1.0 Mb, topografi, router, gateway.

d. Karakteristik Estetika Sistem

Karakteristik umum, operasi, dan fisik adalah parameter kinerja yang objektif. Namun, bagaimana dengan karakteristik subjektif? Disebut juga sebagai karakteristik estetika sistem karena mereka berhubungan dengan "tampilan dan nuansa" dari suatu sistem. Jelas, ini termasuk psikologis, sosiologis, dan perspektif budaya yang terkait dengan daya tarik untuk preferensi Pengguna, Acquirer, atau Pemilik Sistem. Dengan demikian, beberapa pembeli membuat keputusan independen, sementara yang lain dipengaruhi oleh eksternal sistem (yaitu, pembeli lain) dalam hal-hal yang berkaitan dengan komunitas atau status perusahaan, citra, dan kesukaan.

Definisi tentang sistem menunjukkan beberapa karakteristik yang ada di semua sistem, antara lain: organisasi (urutan), interaksi, saling ketergantungan, integrasi, dan tujuan sentral.

1) Organisasi

Organisasi menyiratkan struktur dan ketertiban. Ini adalah pengaturan komponen yang membantu mencapai tujuan. Dalam desain sistem bisnis, misalnya, hubungan hierarki yang dimulai dengan presiden di atas dan mengarah ke bawah hingga pekerja kerah biru merepresentasikan struktur organisasi. Pengaturan seperti itu menggambarkan hubungan sistem-subsistem, mendefinisikan struktur otoritas, menentukan aliran formal komunikasi dan memformalkan rantai komando. Seperti bijaksana, sistem komputer dirancang di sekitar perangkat input, unit pemrosesan pusat, perangkat output dan satu atau lebih unit penyimpanan. Ketika dihubungkan bersama, mereka bekerja sebagai satu kesatuan sistem untuk menghasilkan informasi.

2) Interaksi

Interaksi mengacu pada cara setiap komponen berfungsi dengan komponen lain dari sistem. Dalam sebuah organisasi, misalnya, pembelian harus berinteraksi dengan produksi, periklanan dengan penjualan, dan penggajian dengan personel. Dalam sistem komputer, unit pengolah pusat harus berinteraksi dengan perangkat input untuk memecahkan masalah. Pada gilirannya, memori utama menyimpan program dan data yang digunakan unit aritmatika untuk komputasi. Keterkaitan antara komponen ini memungkinkan komputer untuk bekerja.

3) Saling ketergantungan

Saling ketergantungan berarti bahwa bagian-bagian dari organisasi atau sistem komputer saling bergantung satu sama lain. Mereka dikoordinasikan dan dihubungkan bersama menurut sebuah rencana. Satu subsistem bergantung pada masukan dari subsistem lain agar dapat berfungsi dengan benar: yaitu, keluaran dari satu subsistem adalah masukan yang diperlukan untuk subsistem lainnya. Saling ketergantungan ini sangat penting dalam kerja sistem.

Sistem informasi terintegrasi dirancang untuk melayani kebutuhan pengguna yang berwenang (kepala departemen, manajer, dll.) Untuk akses dan pengambilan cepat melalui terminal jarak jauh. Saling

ketergantungan antara subsistem personel dan pengguna organisasi terlihat jelas.

Singkatnya, tidak ada subsistem yang dapat berfungsi secara terpisah karena bergantung pada data (input) yang diterimanya dari subsistem lain untuk melakukan tugas yang diperlukan. Saling ketergantungan selanjutnya diilustrasikan oleh aktivitas dan dukungan dari analis sistem, pemrogram, dan staf operasi di pusat komputer. Keputusan untuk mengkomputerisasi aplikasi dimulai oleh pengguna, dianalisis dan dirancang oleh analis, diprogram dan diuji oleh pemrogram, dan dijalankan oleh operator komputer. Tak satu pun orang dapat melakukan properti tanpa masukan yang diperlukan dari orang lain di subsistem pusat komputer.

4) Integrasi

Integrasi mengacu pada holisme sistem. Sintesis mengikuti analisis untuk mencapai tujuan utama organisasi. Integrasi berkaitan dengan bagaimana suatu sistem terikat bersama. Ini lebih dari sekadar berbagi bagian atau lokasi fisik. Ini berarti bahwa bagian-bagian dari sistem bekerja bersama di dalam sistem meskipun setiap bagian menjalankan fungsinya yang unik. Integrasi yang berhasil biasanya akan menghasilkan efek sinergis dan dampak total yang lebih besar daripada jika setiap komponen bekerja secara terpisah.

5) Tujuan Utama

Karakteristik terakhir dari suatu sistem adalah tujuan utamanya. Tujuan mungkin nyata atau dinyatakan. Meskipun tujuan yang dinyatakan mungkin merupakan tujuan yang sebenarnya, tidak jarang organisasi menyatakan satu tujuan dan beroperasi untuk mencapai tujuan yang lain. Poin pentingnya adalah bahwa pengguna harus mengetahui tujuan utama dari aplikasi komputer sejak awal analisis untuk desain dan konversi yang sukses. Pertimbangan politik dan organisasi seringkali mengaburkan tujuan sebenarnya. Ini berarti bahwa analis harus mengatasi hambatan tersebut untuk mengidentifikasi tujuan sebenarnya dari perubahan yang diusulkan.

4. Klasifikasi Sistem

Kerangka acuan di mana seseorang memandang suatu sistem terkait dengan penggunaan pendekatan sistem untuk analisis. Sistem telah diklasifikasikan dengan cara yang berbeda. Klasifikasi umum antara lain adalah: (1) fisik atau abstrak, (2) terbuka atau tertutup, dan (3) sistem informasi “buatan manusia”.

a. Sistem fisik atau abstrak

Sistem fisik adalah entitas berwujud yang mungkin statis atau dinamis dalam operasi. Sebagai contoh, bagian fisik pusat komputer adalah petugas, meja, dan kursi yang memudahkan pengoperasian komputer. Mereka bisa dilihat dan dihitung; mereka statis. Sebaliknya, komputer yang diprogram adalah sistem dinamis. Data, program, keluaran, dan aplikasi berubah sesuai permintaan pengguna atau prioritas informasi yang diminta berubah.

Sistem abstrak adalah entitas konseptual atau non-fisik. Mereka mungkin sesederhana rumus hubungan antara set variabel atau model konseptualisasi abstrak dari situasi fisik. Model adalah representasi dari sistem nyata atau terencana. Penggunaan model memudahkan analisis untuk memvisualisasikan hubungan dalam sistem yang diteliti. Tujuannya adalah untuk menunjukkan elemen penting dan keterkaitan kunci dari sistem yang kompleks.

b. Sistem Terbuka atau Tertutup

Klasifikasi lain dari sistem didasarkan pada tingkat kemandiriannya. Sistem terbuka memiliki banyak antarmuka dengan lingkungannya. Ini memungkinkan interaksi melintasi batasnya; ia menerima masukan dari dan mengirimkan keluaran ke luar. Sistem informasi termasuk dalam kategori ini, karena harus beradaptasi dengan perubahan tuntutan pengguna. Sebaliknya, sistem tertutup diisolasi dari pengaruh lingkungan. Pada kenyataannya, sistem yang sepenuhnya tertutup jarang terjadi. Dalam analisis sistem, organisasi, aplikasi, dan komputer selalu terbuka, sistem dinamis dipengaruhi oleh lingkungannya.

Fokus pada karakteristik sistem terbuka sangat tepat waktu mengingat masalah bisnis saat ini dengan penipuan komputer, pelanggaran privasi, kontrol keamanan, dan etika dalam komputasi. Sedangkan aspek teknis dari

analisis sistem berhubungan dengan rutinitas internal dalam area aplikasi pengguna, analisis sistem sebagai sistem terbuka cenderung memperluas cakupan analisis untuk hubungan antara area pengguna dan pengguna lain dan faktor lingkungan yang harus dipertimbangkan sebelum yang baru. sistem akhirnya disetujui. Lebih lanjut, bersikap terbuka terhadap saran menyiratkan bahwa analisis harus fleksibel dan sistem yang dirancang harus responsif terhadap perubahan kebutuhan pengguna dan lingkungan.

Lima karakteristik penting dari sistem terbuka dapat diidentifikasi sebagai berikut:

- 1) **Masukan dari luar:** Sistem terbuka menyesuaikan diri dan mengatur sendiri. Saat berfungsi dengan baik, sistem terbuka mencapai kondisi stabil atau keseimbangan. Dalam perusahaan ritel, misalnya, kondisi mapan terjadi ketika barang dibeli dan dijual tanpa stok atau kelebihan stok. Kenaikan harga pokok memaksa kenaikan harga yang sebanding atau penurunan biaya operasi. Ini respons memberi perusahaan kondisi mapannya.
- 2) **Entropi:** Semua sistem dinamis cenderung menurun seiring waktu, mengakibatkan entropi atau hilangnya energi. Sistem terbuka menolak entropi dengan mencari input baru atau memodifikasi proses untuk kembali ke kondisi mapan. Dalam contoh kita, tidak ada reaksi terhadap kenaikan harga barang dagangan yang membuat bisnis tidak menguntungkan yang dapat memaksanya menjadi bangkrut - keadaan disorganisasi.
- 3) **Proses, keluaran, dan siklus:** Sistem terbuka menghasilkan keluaran yang berguna dan beroperasi dalam siklus, mengikuti jalur aliran kontinu.
- 4) **Diferensiasi:** Sistem terbuka memiliki kecenderungan ke arah peningkatan spesialisasi fungsi dan diferensiasi komponen yang lebih besar. Dalam bisnis, peran orang dan mesin cenderung ke arah spesialisasi yang lebih besar dan interaksi yang lebih besar. Karakteristik ini menawarkan alasan yang kuat untuk meningkatkan nilai konsep sistem dalam pemikiran analisis sistem.
- 5) **Ekuifinalitas:** Istilah ini menyiratkan bahwa tujuan dicapai melalui tindakan yang berbeda dan jalur yang berbeda. Dalam kebanyakan sistem, ada lebih banyak konsensus tentang tujuan daripada pada jalur

untuk mencapai tujuan.

Memahami karakteristik sistem membantu analis untuk mengidentifikasi peran mereka dan menghubungkan aktivitas mereka dengan pencapaian tujuan perusahaan saat mereka melakukan proyek sistem. Analis sendiri adalah bagian dari organisasi. Mereka memiliki peluang untuk menyesuaikan organisasi dengan perubahan melalui aplikasi terkomputerisasi sehingga sistem tidak "rusak". Kunci dari proses ini adalah umpan balik informasi dari pengguna utama sistem baru serta dari manajemen puncak.

Tema proses perancangan sistem informasi banyak meminjam dari pengetahuan umum tentang teori sistem. Tujuannya adalah membuat sistem lebih efisien dengan memodifikasi tujuannya atau mengubah keluarannya.

c. Sistem Informasi Buatan Manusia

Informasi dapat mengurangi ketidakpastian akan suatu keadaan ataupun peristiwa. Misalnya, informasi bahwa dilaut angin sedang tenang dapat mengurangi keraguan bahwa perjalanan perahu akan menyenangkan. Sistem informasi merupakan dasar interaksi antara pengguna dan system analis. Ini memberikan instruksi, perintah dan *feedback*. Ini menentukan sifat hubungan di antara pemangku kebijakan. Bahkan, bisa dipandang sebagai pusat keputusan bagi setiap orang di semua *level*. Dari dasar ini, sistem informasi dapat didefinisikan sebagai seperangkat prosedur dan sistem Operasi yang dirancang di sekitar kriteria berbasis pengguna untuk menghasilkan informasi dan mengkomunikasikannya kepada pengguna untuk perencanaan, pengendalian dan kinerja. Dalam analisis sistem, penting untuk diingat bahwa mempertimbangkan sistem alternatif berarti meningkatkan satu atau lebih kriteria ini.

5. Pendekatan dan Metodologi Pengembangan Sistem

Ada banyak metodologi pengembangan sistem yang berbeda, dan masing-masing unik, berdasarkan urutan dan fokusnya ditempatkan pada setiap fase SDLC. Beberapa metodologi adalah standar formal yang digunakan oleh instansi pemerintah, sedangkan yang lain telah dikembangkan oleh perusahaan konsultan untuk dijual kepada klien. Banyak organisasi memiliki

metodologi internal itu telah diasah selama bertahun-tahun, dan mereka menjelaskan dengan tepat bagaimana setiap fase SDLC dilakukan di perusahaan itu.

Faktor penting lainnya dalam mengkategorikan metodologi adalah pengurutan SDLC fase dan jumlah waktu dan upaya yang dicurahkan untuk masing-masing. Pada hari-hari awal komputasi, programmer tidak memahami perlunya metode siklus hidup yang formal dan terencana dengan baik. Mereka cenderung beralih langsung dari tahap perencanaan yang sangat sederhana ke dalam konstruksi tahap implementasi dengan kata lain, dari yang sangat kabur, tidak baik permintaan sistem pemikiran ke dalam menulis kode. Ini adalah pendekatan yang sama yang terkadang Anda lakukan digunakan saat menulis program untuk kelas pemrograman. Dapat bekerja untuk program kecil itu hanya memerlukan satu programmer, tetapi jika persyaratannya rumit atau tidak jelas, Anda mungkin saja kehilangan aspek penting dari masalah dan harus memulai dari awal lagi, membuang sebagian program (dan waktu serta tenaga yang dihabiskan untuk menulisnya). Pendekatan ini juga membuat kerja tim sulit karena anggota memiliki sedikit gagasan tentang apa yang perlu diselesaikan dan bagaimana caranya bekerja sama untuk menghasilkan produk akhir. Pada bagian ini, dijelaskan tiga kelas yang berbeda dari metodologi pengembangan sistem: *structured design* (desain terstruktur), *rapid application development* (pengembangan aplikasi cepat), dan pengembangan *agile*.

a. Structured Design (Desain Terstruktur)

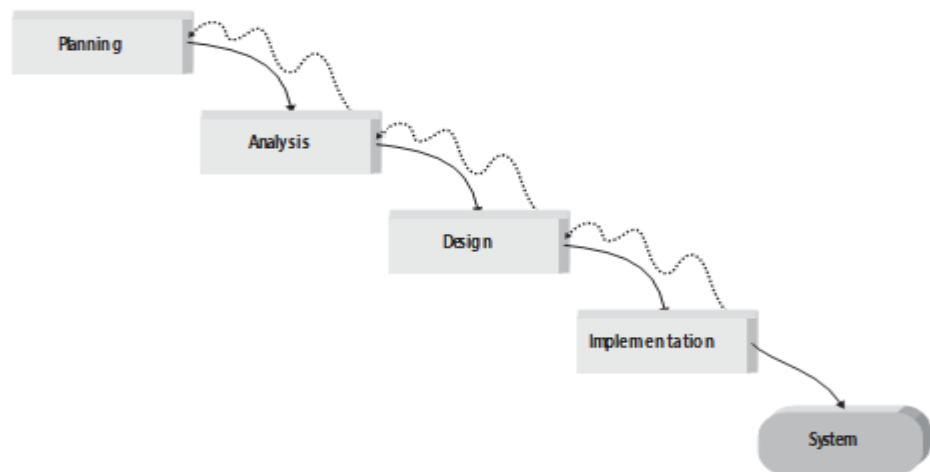
Kategori pertama dari metodologi pengembangan sistem disebut desain terstruktur. Metodologi ini menjadi dominan pada 1980-an, menggantikan ad hoc dan pendekatan yang tidak disiplin. Metodologi *structured design* mengadopsi pendekatan *step by step* dalam siklus pengembangan sistem, berpindah dari satu tahap ke tahap berikutnya secara logis dan beraturan. Berikut adalah dua metodologi yang termasuk dalam *structured design*:

1) Metodologi Pengembangan Waterfall

Metodologi desain terstruktur asli (masih digunakan sampai sekarang) adalah pengembangan *waterfall*. Dengan menggunakan pendekatan pengembangan berbasis *waterfall*, pengembangan sistem dilakukan secara berurutan dari satu tahap ke tahap berikutnya (lihat

Gambar 1). Kiriman utama dari setiap tahap biasanya sangat panjang (biasanya ratusan halaman) dan dikirimkan ke sponsor proyek untuk persetujuan saat proyek berpindah dari satu tahap ke tahap berikutnya.

Desain terstruktur juga memperkenalkan penggunaan pemodelan formal atau teknik diagram untuk menggambarkan proses bisnis dasar dan data yang mendukungnya. Tradisional desain terstruktur menggunakan satu set diagram untuk merepresentasikan proses dan satu set terpisah diagram untuk merepresentasikan data. Karena dua set diagram digunakan, analis sistem harus memutuskan himpunan mana yang akan dikembangkan pertama kali dan digunakan sebagai inti dari sistem: diagram model proses atau diagram model data.



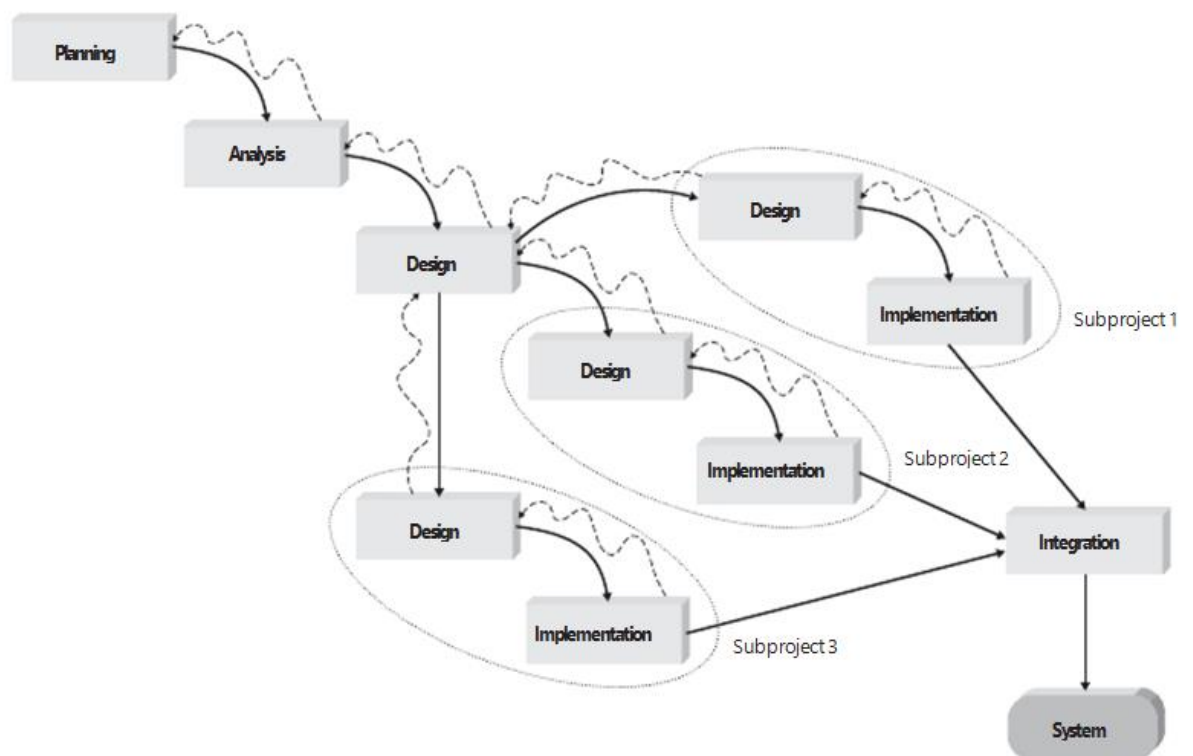
Gambar 1 Metodologi Pengembangan Waterfall

Dua keuntungan utama dari pendekatan waterfall adalah memenuhi semua persyaratan sistem jauh sebelum dimulainya pemrograman dan meminimalkan perubahan pada persyaratan sebagai kelanjutan sebuah proyek. Dua kelemahan utamanya adalah desain harus ditentukan secara lengkap sebelum dimulainya pemrograman dan butuh waktu yang lama antara penyelesaian proposal sistem dan pengiriman sistem ke pengguna (biasanya berbulan-bulan atau bertahun-tahun). Jika tim proyek melewati persyaratan penting, mahal pemrograman pasca implementasi mungkin diperlukan (bayangkan diri Anda mencoba merancang sebuah mobil di kertas;

seberapa besar kemungkinan Anda untuk mengingat lampu interior yang menyala saat pintu buka atau untuk menentukan jumlah katup yang tepat pada mesin?). Sebuah sistem juga bisa membutuhkan pengerjaan ulang yang signifikan karena lingkungan bisnis telah berubah dari waktu ketika fase analisis terjadi.

2) Metodologi Pengembangan Paralel

Pendekatan pengembangan paralel berusaha untuk mengatasi penundaan yang lama antara fase analisis dan pengiriman sistem. Alih-alih merancang dan mengimplementasikannya secara berurutan, ia merancang seluruh sistem dan kemudian membagi proyek menjadi serangkaian subproyek berbeda yang dapat dirancang dan dilaksanakan secara paralel. Setelah semua subproyek selesai, bagian-bagian yang terpisah diintegrasikan dan sistem disampaikan (lihat Gambar 2).



Gambar 2 Metodologi Pengembangan Paralel

Keuntungan utama dari metodologi ini adalah dapat mengurangi waktu pengiriman sistem; dengan demikian, kecil kemungkinan terjadinya perubahan yang menyebabkan pengerjaan ulang. Namun, terkadang subproyek tidak sepenuhnya independen; keputusan desain dibuat dalam satu subproyek dapat mempengaruhi yang lain, dan akhir proyek dapat membutuhkan signifikan upaya integrasi.

b. Rapid Application Development (RAD)

Kategori kedua dari metodologi adalah metodologi pengembangan aplikasi cepat / *Rapid Application Development* (RAD). Metodologi ini merupakan kelas terbaru dari metodologi pengembangan sistem yang muncul di tahun 1990-an. Metodologi ini berupaya untuk mengatasi dua kekurangan metodologi perancangan terstruktur dengan menyesuaikan tahapan SDLC, sehingga bagian-bagian tertentu dari sistem dapat berkembang dengan cepat dan menjangkau pengguna. Dengan cara ini, pengguna dapat lebih memahami sistem dan mengusulkan amandemen untuk membuat sistem lebih dekat dengan konten yang dibutuhkan. Namun, pendekatan berbasis RAD mungkin memiliki masalah kecil: mengelola ekspektasi pengguna. Karena penggunaan alat dan teknologi yang dapat meningkatkan kecepatan dan kualitas pengembangan sistem, ekspektasi pengguna dapat berubah secara dramatis. Sebagai pengguna yang lebih memahami teknologi informasi (TI), persyaratan sistem cenderung berkembang. Ini bukan masalah saat menggunakan metodologi yang menghabiskan banyak waktu untuk mendokumentasikan persyaratan secara menyeluruh.

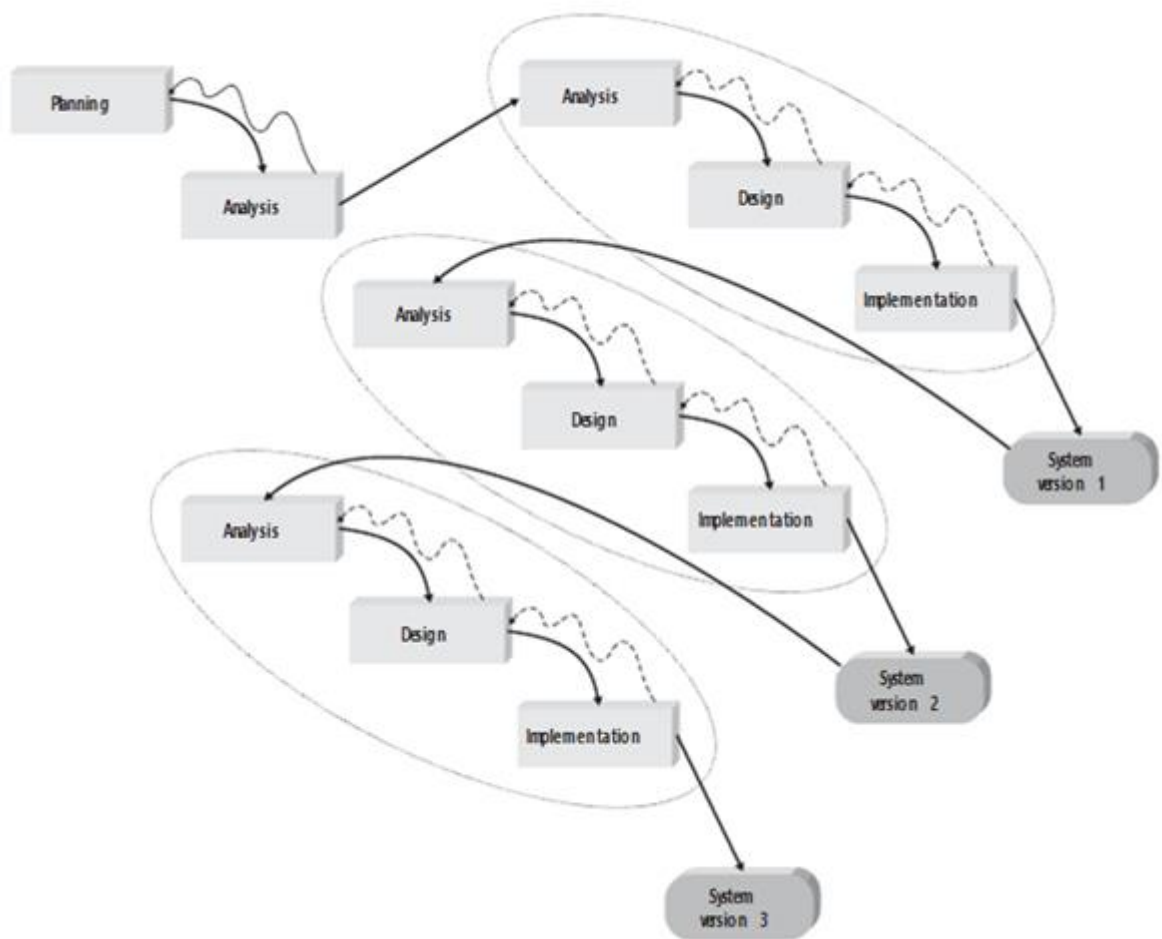
1) Metodologi Pengembangan Bertahap

Berdasarkan metode pengembangan inkremental, seluruh sistem diuraikan menjadi serangkaian versi pengembangan sekuensial. Fase analisis menentukan konsep sistem secara keseluruhan, dan kemudian tim proyek, pengguna, dan sponsor sistem mengklasifikasikan persyaratan ke dalam serangkaian versi. Persyaratan paling penting dan dasar telah dimasukkan ke dalam versi pertama sistem. Kemudian, tahap analisis akan dirancang dan diimplementasikan, tetapi hanya seperangkat persyaratan yang akan diidentifikasi untuk versi satu (lihat

Gambar 3).

Setelah versi satu diimplementasikan, pekerjaan dimulai pada versi dua. Berdasarkan persyaratan yang diidentifikasi sebelumnya serta ide dan masalah baru yang disebabkan oleh pengalaman pengguna versi satu, analisis tambahan dilakukan. Versi dua kemudian dirancang dan diimplementasikan, dan bekerja segera dimulai pada versi berikutnya. Proses ini berlanjut hingga sistem selesai atau tidak lagi digunakan.

Metodologi berbasis pengembangan bertahap memiliki keuntungan untuk segera mendapatkan sistem yang berguna ke tangan pengguna. Meskipun sistem tidak menjalankan semua fungsi yang dibutuhkan pengguna pada awalnya, sistem mulai memberikan nilai bisnis lebih cepat daripada jika sistem dikirimkan setelah selesai, seperti halnya dengan metodologi waterfall dan paralel. Demikian juga, karena pengguna mulai bekerja dengan sistem lebih cepat, mereka cenderung mengidentifikasi persyaratan tambahan penting lebih cepat daripada dengan situasi desain terstruktur.



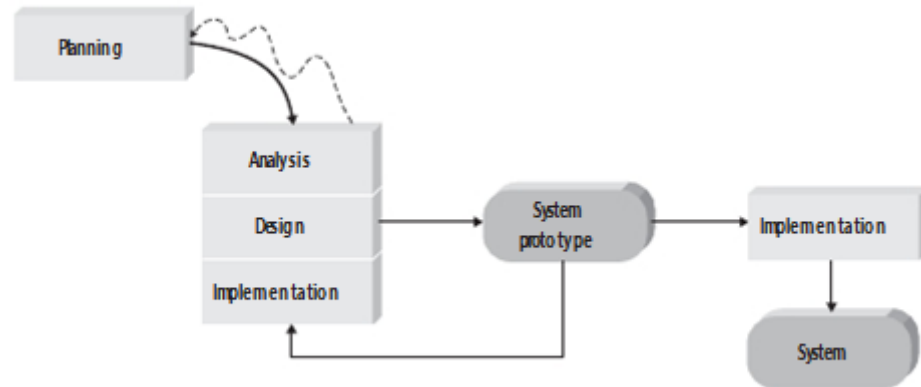
Gambar 3 Metodologi Pengembangan Bertahap

Kelemahan utama dari pengembangan bertahap adalah bahwa pengguna mulai bekerja dengan sistem yang sengaja tidak lengkap. Sangat penting untuk mengidentifikasi fitur yang paling penting dan berguna dan menyertakannya di versi pertama dan untuk mengelola ekspektasi pengguna selama prosesnya.

2) Metodologi Prototipe

Dalam metodologi prototipe tahap analisa, desain, dan implementasi dikerjakan secara bersamaan, dan dilakukan pengulangan pada ketiga tahap tersebut hingga didapatkan sistem yang sesuai dengan keinginan. Dengan menggunakan metode ini, dapat dibuat dasar analisa dan desain, dan segera mulai mengerjakan prototipe sistem, yang merupakan program cepat dan kotor yang hanya

menyediakan fungsionalitas minimal. Setelah prototipe (sekarang disebut "sistem") dipasang, perbaikan dilakukan hingga diterima sebagai sistem baru (lihat Gambar 4).



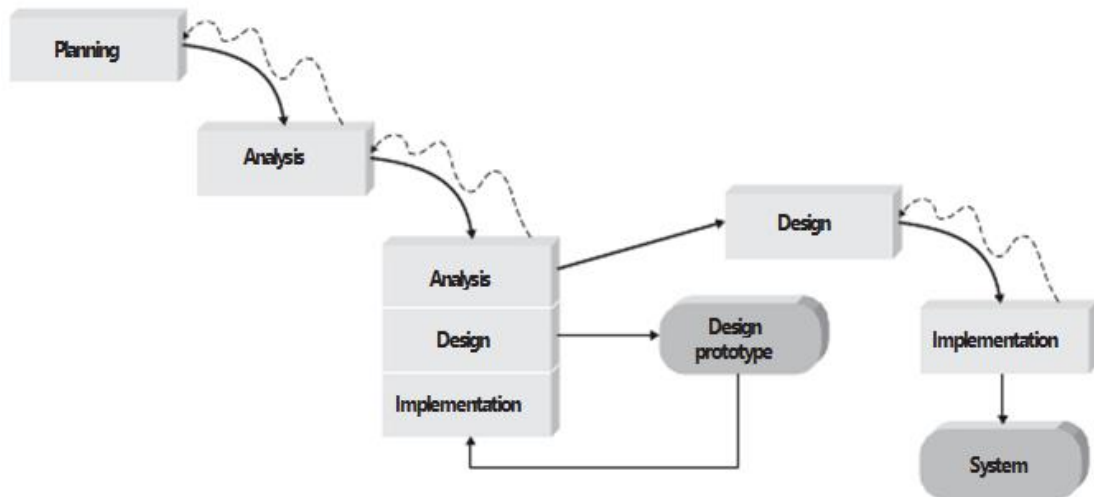
Gambar 4 Metodologi Prototipe

Keuntungan utama dari metodologi berbasis prototipe adalah bahwa metodologi ini sangat cepat menyediakan sistem yang dapat digunakan pengguna untuk berinteraksi, bahkan jika belum siap untuk digunakan organisasi secara luas pada awalnya. pembuatan prototipe meyakinkan pengguna bahwa tim proyek sedang mengerjakan sistem (tidak ada penundaan besar ketika pengguna melihat sedikit kemajuan), dan pembuatan prototipe membantu menyesuaikan kebutuhan nyata dengan lebih cepat.

Masalah utama dengan pembuatan prototipe adalah bahwa sistemnya yang bergerak cepat melepaskan tantangan upaya untuk melakukan analisis metodis yang cermat. Seringkali prototipe mengalami perubahan signifikan sehingga banyak keputusan desain awal menjadi salah.

3) Metodologi Prototipe Sekali Pakai

Metodologi berbasis prototipe sekali pakai mirip dengan metodologi berbasis prototipe yang mencakup pengembangan prototipe; namun, prototipe sekali pakai dilakukan pada titik yang berbeda di SDLC. Prototipe ini digunakan untuk tujuan yang sangat berbeda dari yang telah dibahas sebelumnya, dan memiliki tampilan yang sangat berbeda (lihat Gambar 5).



Gambar 5. Metodologi Prototipe Sekali Pakai

Metodologi yang didasarkan pada prototipe sekali pakai memiliki tahap analisis yang relatif komprehensif, yang digunakan untuk mengumpulkan informasi dan mengembangkan ide untuk konsep sistem.

Namun, pengguna mungkin tidak sepenuhnya memahami banyak fitur yang mereka sarankan, dan mungkin ada masalah teknis yang sulit untuk diselesaikan. Masing-masing masalah ini diselidiki dengan menganalisis, merancang, dan membangun prototipe. Prototipe desain bukanlah sistem yang berfungsi. Ini adalah produk yang merupakan bagian dari sistem yang membutuhkan perbaikan tambahan dan hanya berisi detail yang cukup bagi pengguna untuk memahami masalah yang dihadapi. Misalnya, pengguna tidak sepenuhnya jelas tentang cara kerja sistem entri pesanan.

Sebuah sistem yang dikembangkan menggunakan metodologi jenis ini mengandalkan beberapa prototipe desain selama tahap analisis dan desain. Setiap prototipe digunakan untuk meminimalkan risiko yang terkait dengan sistem dengan memastikan bahwa masalah penting dipahami sebelum yang sebenarnya sistem dibangun. Setelah masalah terselesaikan, proyek beralih ke desain dan implementasi. Pada titik ini, prototipe desain tidak digunakan lagi, yang merupakan perbedaan penting antara metodologi lain dan metodologi pembuatan prototipe ini, di mana prototipe berkembang menjadi sistem akhir.

Metodologi berbasis prototipe sekali pakai menyeimbangkan manfaat fase analisis dan desain yang dipikirkan matang-matang dengan keuntungan menggunakan prototipe untuk memperbaiki masalah utama sebelum sistem dibangun. Diperlukan waktu lebih lama untuk menyampaikan sistem akhir dibandingkan dengan metodologi berbasis prototipe, tetapi jenis metodologi ini biasanya menghasilkan sistem yang lebih stabil dan andal.

c. Pengembangan Agile

Kategori ketiga dari metodologi pengembangan sistem masih muncul saat ini: pengembangan agile. Semua metodologi pengembangan agile didasarkan pada manifesto tangkas dan seperangkat dua belas prinsip. Penekanan dari manifesto adalah untuk memfokuskan pengembang pada kondisi kerja pengembang, perangkat lunak yang berfungsi, pelanggan, dan menangani persyaratan yang berubah alih-alih berfokus pada proses pengembangan sistem terperinci, alat, semua dokumentasi inklusif, kontrak hukum, dan rencana terperinci. Metodologi yang berpusat pada pemrograman ini memiliki sedikit aturan dan praktik, yang semuanya cukup mudah diikuti. Metodologi ini biasanya hanya didasarkan pada dua belas prinsip perangkat lunak tangkas. Prinsip-prinsip ini termasuk yang berikut ini:

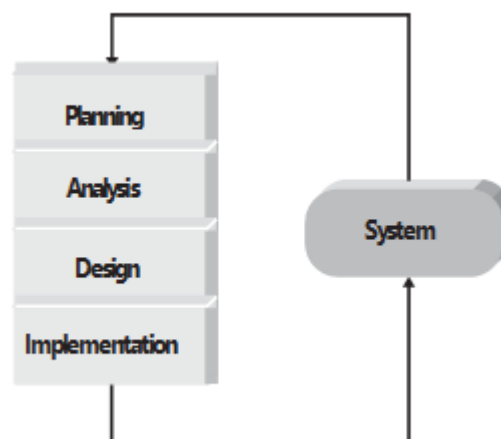
- 1) Perangkat lunak dikirimkan lebih awal dan terus menerus melalui proses pengembangan, untuk memuaskan pelanggan.
- 2) Persyaratan yang berubah diterapkan terlepas dari kapan hal itu terjadi dalam proses pengembangan.
- 3) Perangkat lunak yang berfungsi sering dikirimkan ke pelanggan.
- 4) Pelanggan dan pengembang bekerja sama untuk memecahkan masalah bisnis.
- 5) Individu yang termotivasi menciptakan solusi; memberi mereka alat dan lingkungan yang mereka butuhkan, dan percayai mereka untuk memberikannya.
- 6) Komunikasi tatap muka dalam tim pengembangan adalah metode pengumpulan persyaratan yang paling efisien dan efektif.
- 7) Ukuran utama kemajuan bekerja, menjalankan perangkat lunak.
- 8) *User* maupun pengembang harus bekerja dengan kecepatan yang berkelanjutan. Artinya, tingkat pekerjaan bisa dipertahankan tanpa batas

waktu tanpa ada pekerja yang mengalami kejenuhan.

- 9) Ketangkasan ditingkatkan melalui perhatian pada keunggulan teknis dan desain yang baik.
- 10) Kesederhanaan, penting untuk menghindari pekerjaan yang tidak perlu.
- 11) Tim mengembangkan arsitektur, persyaratan, dan desain terbaik.
- 12) Pengembang secara teratur merefleksikan bagaimana meningkatkan proses pengembangan mereka.

Berdasarkan prinsip-prinsip ini, metodologi agile berfokus pada penyederhanaan proses pengembangan sistem dengan meniadakan banyak overhead pemodelan dan dokumentasi serta waktu yang dibutuhkan untuk tugas-tugas tersebut. Sebaliknya, proyek tersebut menitikberatkan pada pengembangan aplikasi berulang yang sederhana.

Semua metodologi pengembangan agile mengikuti siklus sederhana melalui fase tradisional dari proses pengembangan sistem (lihat Gambar 6). Hampir semua metodologi agile digunakan dalam hubungannya dengan teknologi berorientasi objek.



Gambar 6. Pengembangan Agile

Namun, metodologi agile memang mendapat kritik. Salah satu kritik utama berkaitan dengan lingkungan bisnis, di mana sebagian besar pengembangan sistem informasi aktual di luar batas, dialihdayakan, dan / atau disubkontrakkan. Mengingat metodologi pengembangan agile yang membutuhkan lokasi bersama tim pengembangan, ini tampaknya merupakan asumsi yang sangat tidak realistis. Kritik besar kedua adalah

bahwa jika pengembangan tangkas tidak dikelola dengan hati-hati, dan menurut definisi tidak, proses pengembangan dapat beralih ke pendekatan prototipe yang pada dasarnya menjadi lingkungan "programmer menjadi liar" di mana programmer mencoba meretas solusi bersama. Kritik utama ketiga, yang didasarkan pada kurangnya dokumentasi aktual yang dibuat selama pengembangan perangkat lunak, menimbulkan masalah terkait kemampuan audit dari sistem yang dibuat. Tanpa dokumentasi yang memadai, baik sistem maupun proses pengembangan sistem tidak dapat dijamin. Kritik besar keempat didasarkan pada apakah pendekatan tangkas dapat menghasilkan sistem misi kritis yang besar.

Bahkan dengan kritik ini, mengingat potensi pendekatan tangkas untuk mengatasi backlog aplikasi dan memberikan solusi tepat waktu untuk banyak masalah bisnis, pendekatan tangkas harus dipertimbangkan dalam beberapa keadaan. Selain itu, banyak teknik yang didorong dengan memperhatikan tujuan yang mendasari manifesto tangkas dan sekumpulan dua belas prinsip tangkas sangat berguna dalam pengembangan sistem berorientasi objek. Dua dari contoh metodologi pengembangan agile yang lebih populer adalah program ekstrim (XP) dan Scrum.

1) Pemrograman Ekstrim/ Extreme Programming (XP)

Pemrograman Ekstrim memiliki empat nilai inti antara lain: komunikasi, kesederhanaan, *feedback*, dan keberanian. Keempat nilai ini merupakan dasar bagi pengembang XP untuk membangun sistem apa pun. Pertama, pengembang harus terus memberikan *feedback* kepada pengguna. Kedua, XP mengharuskan pengembang mengikuti prinsip KISS. Ketiga, pengembang harus dapat melakukan perubahan tambahan untuk memperluas sistem. Tidak hanya harus melakukan perubahan, mereka juga harus menerima perubahan. Keempat, developer harus memiliki pola pikir yang mengutamakan kualitas. XP juga mendukung anggota tim untuk mengembangkan keterampilan mereka.

Tiga prinsip utama keberhasilan penggunaan sistem XP adalah pengujian berkelanjutan, pengkodean sederhana yang dilakukan oleh sepasang pengembang, dan interaksi erat dengan pengguna akhir untuk membangun sistem dengan sangat cepat. Pengujian dan coding yang efisien adalah inti dari XP. Kode diuji setiap hari dan ditempatkan ke

dalam lingkungan pengujian integratif. Jika ada bug, kode tersebut dicadangkan hingga benar-benar bebas dari kesalahan.

Proyek XP dimulai dengan cerita pengguna yang menjelaskan apa yang perlu dilakukan sistem. Kemudian, programmer membuat kode dalam modul kecil dan sederhana dan menguji untuk memenuhi kebutuhan tersebut. Pengguna diharuskan tersedia untuk menjernihkan pertanyaan dan masalah yang muncul. Standar sangat penting untuk meminimalkan kebingungan, jadi tim XP menggunakan sekumpulan nama, deskripsi, dan praktik pengkodean yang umum. Proyek XP memberikan hasil lebih cepat daripada pendekatan RAD, dan mereka jarang macet dalam mengumpulkan persyaratan untuk sistem.

Penganut XP mengklaim banyak kekuatan yang terkait dengan pengembangan perangkat lunak menggunakan XP. Programmer bekerja erat dengan semua pemangku kepentingan, dan komunikasi di antara semua pemangku kepentingan ditingkatkan. Pengujian berkelanjutan dari sistem yang berkembang didorong. Sistem ini dikembangkan secara evolusioner dan bertahap, yang memungkinkan persyaratan berkembang seiring dengan pemahaman para pemangku kepentingan terhadap potensi yang dimiliki teknologi dalam memberikan solusi untuk masalah mereka. Estimasi didorong oleh tugas dan dilakukan oleh programmer yang akan mengimplementasikan solusi untuk tugas yang sedang dipertimbangkan. Karena semua pemrograman dilakukan berpasangan, tanggung jawab bersama untuk setiap komponen perangkat lunak berkembang di antara programmer. Akhirnya, kualitas produk akhir meningkat selama setiap iterasi.

Untuk proyek kecil dengan tim yang sangat termotivasi, kohesif, stabil, dan berpengalaman, XP seharusnya berfungsi dengan baik. Namun, jika proyeknya tidak kecil atau timnya tidak senang, keberhasilan upaya pengembangan XP diragukan. Hal ini cenderung meragukan seluruh gagasan membawa kontraktor luar ke lingkungan tim yang ada menggunakan XP. Kemungkinan orang luar bercengkerama dengan orang dalam mungkin terlalu optimis. XP membutuhkan disiplin yang tinggi, jika tidak, proyek akan menjadi tidak fokus dan kacau. XP disarankan hanya untuk sekelompok kecil pengembang (tidak lebih dari sepuluh pengembang) dan tidak

disarankan untuk aplikasi penting yang besar. Karena kurangnya dokumentasi analisis dan desain, hanya ada dokumentasi kode yang terkait dengan XP, jadi memelihara sistem besar yang dibangun dengan XP mungkin mustahil. Dan karena sistem informasi bisnis mission-critical cenderung ada untuk waktu yang lama, kegunaan XP sebagai metodologi pengembangan sistem informasi bisnis diragukan. Akhirnya, metodologi membutuhkan banyak masukan pengguna di tempat, sesuatu yang tidak dapat dilakukan oleh banyak unit bisnis. Namun, beberapa teknik yang terkait dengan XP berguna dalam pengembangan sistem berorientasi objek. Misalnya, cerita pengguna, pemrograman pasangan, dan pengujian berkelanjutan adalah alat yang sangat berharga dari mana pengembangan sistem berorientasi objek bisa mendapatkan keuntungan.

2) Scrum

Scrum adalah istilah yang sangat dikenal oleh para penggemar rugby. Dalam rugby, scrum digunakan untuk memulai kembali permainan. Singkatnya, pencipta metode *scrum* percaya bahwa tidak peduli seberapa banyak Anda merencanakan, segera setelah perangkat lunak mulai dikembangkan, kekacauan muncul dan rencana tersebut keluar dari jendela. Hal terbaik yang dapat Anda lakukan adalah bereaksi terhadapnya. Di mana bola rugby pepatah menyemprot keluar. Anda kemudian melakukan sprint dengan bola hingga *scrum* berikutnya. Dalam kasus metodologi Scrum, sprint berlangsung selama tiga puluh hari kerja. Di akhir sprint, sebuah sistem dikirimkan ke pelanggan.

Di permukaan, Scrum adalah metode pengembangan sistem yang paling membingungkan. Untuk mengontrol kekacauan bawaan tertentu, pengembangan Scrum berfokus pada beberapa praktik utama. Tim ini mengatur diri sendiri dan mengarahkan diri sendiri. Tidak seperti metode lain, tim Scrum tidak memiliki ketua tim yang ditunjuk. Sebaliknya, tim mengatur secara simbiosis dan menetapkan tujuan sendiri untuk setiap sprint (iterasi). Setelah sprint dimulai, tim Scrum tidak mempertimbangkan persyaratan tambahan apa pun. Persyaratan baru apa pun yang ditemukan ditempatkan di simpanan persyaratan yang masih perlu ditangani. Di awal setiap hari kerja, pertemuan Scrum berlangsung. Di akhir setiap sprint, tim mendemonstrasikan perangkat

lunak tersebut kepada klien. Berdasarkan hasil sprint, rencana baru dimulai untuk sprint berikutnya.

Rapat scrum adalah salah satu aspek paling menarik dari proses pengembangan Scrum. Anggota tim menghadiri rapat, tetapi siapa pun dapat hadir. Namun, dengan sedikit pengecualian, hanya anggota tim yang boleh berbicara. Satu pengecualian yang menonjol adalah manajemen memberikan umpan balik tentang relevansi bisnis dari pekerjaan yang dilakukan oleh tim tertentu. Dalam pertemuan ini, semua anggota tim berdiri membentuk lingkaran dan melaporkan apa yang telah mereka capai pada hari sebelumnya, menyampaikan apa yang akan dilakukan hari ini, dan menjelaskan apa saja kemajuan yang diblokir hari sebelumnya. Untuk mengaktifkan kemajuan berkelanjutan, setiap blok yang diidentifikasi ditangani dalam waktu satu jam. Dari sudut pandang Scrum, lebih baik membuat keputusan "buruk" tentang suatu blok pada saat ini dalam pengembangan daripada tidak membuat keputusan. Karena pertemuan berlangsung setiap hari, keputusan yang buruk dapat dengan mudah dibatalkan. Larman menyarankan agar setiap anggota tim harus melaporkan persyaratan tambahan apa pun yang telah ditemukan selama sprint dan apa pun yang dipelajari anggota tim yang dapat berguna untuk diketahui anggota tim lainnya.

Salah satu kritik utama dari Scrum, seperti halnya semua metodologi tangkas, adalah bahwa masih dipertanyakan apakah Scrum dapat berkembang untuk mengembangkan sistem misi-kritis yang sangat besar. Ukuran tim Scrum biasanya tidak lebih dari tujuh anggota. Satu-satunya prinsip pengorganisasian yang dikemukakan oleh pengikut Scrum untuk menjawab kritik ini adalah dengan mengatur scrum dari scrum. Setiap tim bertemu setiap hari, dan setelah rapat tim berlangsung, seorang perwakilan (bukan pemimpin) dari setiap tim menghadiri rapat scrum-of-scrums. Ini berlanjut sampai kemajuan seluruh sistem telah ditentukan. Bergantung pada jumlah tim yang terlibat, pendekatan untuk mengelola proyek besar ini diragukan. Namun, seperti di XP dan pendekatan pengembangan tangkas lainnya, banyak ide dan teknik yang terkait dengan pengembangan Scrum berguna dalam pengembangan sistem berorientasi objek, seperti fokus pertemuan Scrum, pendekatan evolusioner dan inkremental untuk

mengidentifikasi persyaratan, dan inkremental. dan pendekatan iteratif untuk pengembangan sistem.

C. SOAL LATIHAN/ TUGAS

1. Carilah definisi pengertian sistem dari para ahli selain yang dijelaskan di atas!
2. Definisikan pengertian sistem menurut pendapatmu!
3. Apa yang kalian ketahui tentang klasifikasi sistem? Jelaskan!
4. Apa yang kalian ketahui tentang metodologi pengembangan sistem? Jelaskan!
5. Jelaskan secara singkat alur metodologi pengembangan sistem!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Charles S. Wasson (2006). *System Analysis, Design, and Development: Concepts, Principles, and Practices*. New Jersey: John Wiley & Sons, Inc.

Dr. Jawahar. *Overview of System Analysis & Design*. Diakses dari: <http://www.ddegjust.ac.in/studymaterial/pgdca/ms-04.pdf>

GLOSARIUM

Application atau Program Aplikasi adalah perangkat lunak yang dikembangkan oleh perusahaan komputer agar dapat menyelesaikan tugas tertentu, seperti Microsoft-Word dan Microsoft-Excel.

Data merupakan kumpulan angka dan karakter yang tidak mempunyai arti. Data tersebut dapat diolah untuk menghasilkan sebuah informasi.

Ethernet adalah Standar perangkat keras LAN (Local Area Network) untuk kabel dan spesifikasi transmisi.

Fitur dari kata *feature*, adalah fungsi atau kemampuan khusus yang ada pada suatu alat.

Hacker atau peretas adalah pakar komputer terampil yang menggunakan pengetahuan teknis mereka untuk memecahkan masalah. Meskipun "peretas" dapat merujuk pada pemrogram komputer yang terampil, istilah ini telah dikaitkan dengan "peretas keamanan" dalam budaya populer, dan "peretas keamanan" mengandalkan pengetahuan teknis mereka untuk menggunakan kesalahan atau celah untuk menyerang sistem komputer.

Java merupakan bahasa pemrograman yang digunakan untuk membuat konten aktif di halaman web, yang juga dapat dijalankan di komputer mana pun.

Komputer adalah peralatan elektronik yang dapat mengolah berbagai data secara sistematis dan cermat. Misalnya data digital, suara dan gambar.

Soft Real-Time merupakan sistem yang bila tidak berhasil diselesaikan dalam waktu satu hari maka tidak menyebabkan kegagalan sistem.

Wizard adalah sebuah fungsi dalam perangkat lunak, termasuk perkantoran, dapat mendesain dokumen dengan mudah dan langkah demi langkah.

PERTEMUAN 2

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian, sejarah pengembangan, tahapan dan alat pengembangan SDLC (*Systems Development Life Cycle*). Dari pertemuan ini diharapkan mahasiswa mampu memahami apa itu SDLC.

B. URAIAN MATERI

1. Pengertian SDLC

Menurut bahasa SDLC bisa diartikan dengan siklus hidup pengembangan sistem. Siklus hidup sistem / produk berfungsi sebagai peta jalan mendasar untuk memahami dan mengkomunikasikan bagaimana sistem alami dan buatan manusia berkembang melalui perkembangan fase siklus hidup yang berurutan. Untuk sistem buatan manusia, peta jalan memberikan dasar untuk menilai kapabilitas dan kinerja sistem yang ada terkait dengan ancaman dan peluang; mendefinisikan, mengadakan, dan mengembangkan sistem baru untuk menanggapi ancaman dan peluang; dan menerapkan sistem baru untuk mencapai tujuan misi yang melawan atau memanfaatkan ancaman dan peluang.

Evolusi sistem apa pun yang dibuat oleh atau diketahui manusia dimulai dari titik pembuahan dan berakhir di pembuangan. Proses ini disebut sebagai siklus hidup sistem. Siklus hidup sistem berfungsi secara struktural sebagai landasan pengembangan sistem. Sistem buatan manusia dikonseptualisasikan, direncanakan, diatur, dijadwalkan, diperkirakan, diadakan, disebarkan, dioperasikan dan didukung, dan dibuang menggunakan struktur ini. Sistem alam mengikuti konstruksi serupa dengan fase kehidupan.

Siklus hidup untuk sistem, produk, atau layanan apa pun terdiri dari serangkaian fase yang dimulai dengan konsepsi sistem dan berlanjut hingga pembuangan akhir. Untuk sistem buatan manusia awalnya dan berakhirnya

setiap fase ditandai dengan titik kontrol yang signifikan atau peristiwa pementasan seperti keputusan kunci pada tinjauan teknis atau acara lapangan yang mengizinkan kemajuan ke fase berikutnya. Siklus hidup sistem yang khas terdiri dari serangkaian fase, antara lain adalah tahap definisi sistem, tahap pengadaan sistem, tahap pengembangan sistem, fase operasi dan dukungan sistem (O&S), tahap produksi sistem, fase pembuangan sistem

Bab ini menyajikan siklus hidup sistem / produk sebagai kerangka kerja tingkat atas dari fase tertanam yang diperlukan untuk mengembangkan kebutuhan operasional Pengguna untuk sistem, produk, atau layanan dari "visi" konseptual melalui pembuangan. Setiap tahapan mewakili sekumpulan aktivitas yang berfokus pada tujuan program dan hasil kerja tertentu. Seperti yang akan segera Anda temukan, beberapa fase ini memiliki akhir yang terdefinisi dengan baik yang ditandai oleh tonggak penting sementara fase lain tumpang tindih dan transisi dari satu fase ke fase lainnya.

2. Sejarah Pengembangan SDLC

Kerangka kerja metodologi pengembangan perangkat lunak (juga dikenal sebagai SDM). Menurut Elliott (2004) siklus hidup pengembangan sistem (SDLC) bisa dianggap sebagai kerangka metodologi formal tertua untuk membangun sistem informasi. Ide awal dari SDLC adalah "untuk mengejar pengembangan sistem informasi dengan cara yang sangat terencana, terstruktur dan metodis, yang membutuhkan setiap tahap siklus hidup dari awal hingga penyampaian sistem akhir dilakukan secara kaku dan berurutan" dalam konteks kerangka yang diterapkan. Sasaran utama dari kerangka metodologi ini pada tahun 1960-an adalah "untuk mengembangkan sistem bisnis fungsional berskala besar. Aktivitas sistem informasi berkisar pada pemrosesan data yang berat dan rutinitas penghitungan angka".

Metodologi, proses, dan kerangka kerja berkisar dari langkah-langkah terlarang khusus yang dapat digunakan secara langsung oleh organisasi dalam pekerjaan sehari-hari, hingga kerangka kerja fleksibel yang digunakan organisasi untuk menghasilkan serangkaian langkah khusus yang disesuaikan dengan kebutuhan proyek tertentu atau kelompok. Dalam beberapa kasus, organisasi "sponsor" atau "pemeliharaan" mendistribusikan sekumpulan dokumen resmi yang menjelaskan proses tersebut. Berikut adalah ringkasan

sejarah pengembangan SDLC dari awal hingga sekarang:

- a. Tahun 1970-an
 - 1) Pemrograman Terstruktur, Sejak 1969
 - 2) Cap Gemini SDM, Sejak 1974
- b. Tahun 1980-an
 - 1) Structured Systems Analysis and Design Method atau disingkat SSADM, Sejak 1980
 - 2) Analisis Kebutuhan Informasi / Metodologi sistem lunak, Sejak 1981
- c. Tahun 1990-an
 - 1) Pemrograman berorientasi objek (OOP) dikembangkan pada awal 1960-an, dan menjadi pendekatan pemrograman yang dominan selama pertengahan 1990-an
 - 2) Pengembangan aplikasi cepat (RAD), sejak 1991
 - 3) Metode pengembangan sistem dinamis (DSDM), sejak 1994
 - 4) Scrum, sejak 1995
 - 5) Proses perangkat lunak tim, sejak 1998
 - 6) Rational Unified Process (RUP), dikelola oleh IBM sejak 1998
 - 7) Pemrograman ekstrim, sejak 1999
- d. Tahun 2000-an
 - 1) Agile Unified Process (AUP) dipertahankan sejak 2005 oleh Scott Ambler
 - 2) Pengiriman tangkas yang disiplin (DAD) Menggantikan AUP
- e. Tahun 2010-an
 - 1) Scaled Agile Framework (SAFe)
 - 2) Scrum Skala Besar (LeSS)
 - 3) DevOps

Ada Lovelace dikenal karena menulis program dasar pertama pada tahun 1843 untuk Mesin Analitik, yang dirancang oleh Charles Babbage. Tetapi hanya setelah bahasa pertama 'short-code' (0 dan 1) muncul pada tahun 1949, dan kompiler pertama ditulis pada tahun 1951; lahirlah program komputer pertama. Pada tahun 1957, Fortran muncul sebagai salah satu bahasa pemrograman besar pertama, yang diikuti oleh Cobol. Dalam 175 tahun sejarah pemrograman hingga saat ini, umat manusia telah mengembangkan lebih dari 700 bahasa pemrograman.

Anehnya, Siklus Hidup Pengembangan Sistem (SDLC) disebut sebagai riwayat Siklus hidup pengembangan *software* yang tidak sedalam program *software*. Kerangka konseptual, "SDLC" yang mempertimbangkan struktur tahapan yang terlibat dalam pengembangan aplikasi dari studi kelayakan awal hingga penerapannya di lapangan dan pemeliharaan, menjadi menonjol dengan model Waterfall. Winston W. Royce mengutip deskripsi formal pertama dari model air terjun dalam sebuah artikel pada tahun 1970.

Model air terjun menyediakan metode yang terorganisir dan terkontrol untuk mengelola proyek; model berkembang secara linier melalui tahapan diskrit, logis dan dapat diinterpretasikan, sehingga mudah untuk dipahami dan diterapkan. Ini memberikan tonggak yang mudah diidentifikasi dalam proses pengembangan. Model air terjun mempertahankan dominasinya selama dua dekade dan masih banyak digunakan oleh banyak organisasi. Perpanjangan formal Waterfall dikembangkan oleh Jerman yang disebut V-model pada tahun 1980 untuk proyek pertahanan, dan sekarang telah ditemukan aplikasi yang luas dalam program komersial serta pertahanan. Model V merangkum langkah-langkah utama yang harus diambil sehubungan dengan kiriman yang sesuai dalam pengembangan siklus hidup proyek.

Saat fokus bergerak ke arah customer centricity, model Waterfall mulai menuai kritik untuk pendekatan bertahap liniernya, yang tidak memungkinkan adanya fleksibilitas untuk perubahan pelanggan di tengah-tengah pengembangan perangkat lunak dan pada banyak kesempatan memiliki garis waktu yang memanjang, yang mengakibatkan waktu yang tinggi untuk -pasar. Pengembangan berulang diciptakan sekitar tahun 1975 sebagai tanggapan atas inefisiensi dan masalah yang ditemukan dalam model Waterfall. Komunitas TI menggembarkan-gemborkannya sebagai terobosan besar, karena model Iteratif berusaha untuk membangun kerangka kerja yang gesit dan adaptif. Hal ini juga membawa revolusi bagi komunitas manajemen proyek. Banyak metode proses pengembangan perangkat lunak baru mulai bermunculan, dan setiap metode telah membawa ide dan konsep baru untuk mengelola proyek, seperti Evolutionary (1976), Incremental (1978), Stage Gate (1983), Spiral (1986), dll. Iterative, Incremental dan Spiral memperoleh apresiasi yang signifikan di antara Manajer Proyek dari berbagai organisasi dan masih digunakan di seluruh dunia oleh banyak orang.

Iterative dan Incremental sering saling melengkapi dan dalam beberapa proyek, keduanya digunakan bersamaan. Fondasi dasar dari metode ini mendukung pengembangan sistem melalui siklus berulang (Iteratif) dan dalam himpunan bagian yang lebih kecil dalam satu contoh (Inkremental), sehingga memungkinkan tim untuk memanfaatkan pembelajaran dari fase sebelumnya dan berimprovisasi dalam iterasi saat ini. Pembelajaran diperoleh baik melalui proses pengembangan maupun penggunaan sistem sub-set (Incremental) yang digunakan. Proses ini biasanya dimulai dengan implementasi subset dari persyaratan perangkat lunak dan secara berulang meningkatkan versi yang berkembang sampai sistem penuh diimplementasikan. Pada setiap iterasi, modifikasi desain dibuat dan kemampuan fungsional baru ditambahkan.

Evolusi SDLC berlanjut, dengan fokus pada ketangkasan dalam upaya untuk mengurangi waktu ke pasar, membangun produk yang layak minimum sambil menjaga pelanggan di pusat segalanya. Ini membutuhkan paradigma perubahan karena mempromosikan individu dan interaksi melalui proses dan alat, perangkat lunak yang bekerja melalui dokumen yang komprehensif, kolaborasi pelanggan melalui semua fase, dan menanggapi perubahan mengikuti rencana. Kebutuhan saat ini adalah untuk memiliki metode yang lebih fleksibel yang memenuhi permintaan. Metode *agile* muncul sebagai spin-off langsung metode perangkat lunak dari tahun 1980-an, yaitu Joint Application Design atau JAD (1986), Rapid Systems Development atau RSD (1987), dan Rapid Application Development atau RAD (1991).

Metode Agile secara resmi dimulai pada 1990-an dan banyak Metode Pengembangan Perangkat Lunak dikembangkan dalam beberapa dekade mendatang. Ini dimulai dengan Crystal (1991) yang berfokus terutama pada orang dan interaksi, Scrum (1993) yang mencakup empat manifesto tangkas, Pengembangan Sistem Dinamis (1994) muncul ke cakrawala setelah manajer proyek menggunakan RAD (Pengembangan Aplikasi Cepat) mencari lebih banyak tata kelola dan disiplin terhadap cara kerja iteratif baru, Synch-Stabilize (1995) menanamkan kerja paralel pada modul aplikasi individu, sering menyinkronkan kode yang dikembangkan dengan yang ada di tim lain, dan secara teratur men-debug (menstabilkan) kode selama proses pengembangan, Feature Driven Development (1996) yang memadukan sejumlah praktik terbaik tangkas yang diakui industri menjadi keseluruhan kohesif yang didorong oleh nilai klien.

Ada berbagai metode agile lainnya seperti Judo Strategy di tahun 1997, Internet Time di tahun 1998, New Development Rhythm di tahun 1989, Adaptive Software Development di tahun 1999, Open Source Software Development di tahun 1999, Lean Development di tahun 2003, dan Agile Proses Terpadu di tahun 2005. Namun, keberhasilan Extreme Programming (1999) menyebabkan adopsi metode *agile* yang belum pernah terjadi sebelumnya pada awal tahun 2000-an.

Extreme Programming (XP) dirancang untuk meningkatkan kualitas *software* dan daya tanggap terhadap perubahan kebutuhan pengguna. Inovasi metode ini mengubah berbagai praktik yang ada hingga ke tingkat yang ekstrim. Misalnya, praktik XP dari Pemrograman Berpasangan mengubah praktik tinjauan kode yang ada ke tingkat yang ekstrem, karena praktik tersebut mendorong kode produksi untuk ditulis oleh dua pengembang di satu mesin (mis. Pengemudi dan navigator). Banyak dari praktik lain dalam Pemrograman Ekstrim membawa perubahan radikal pada metode yang ada, beberapa di antaranya adalah Integrasi Berkelanjutan, permainan perencanaan, pelanggan di lokasi, Refactoring, rilis kecil, dan desain sederhana.

Mengurangi waktu penyelesaian melalui adopsi yang gesit secara inheren membuat departemen Operasi tertinggal dengan penerapan yang menumpuk lebih cepat daripada yang bisa dirilis. Tren ini pada akhirnya mendorong munculnya DevOps, versi baru dari metodologi Agile yang mencakup segmen Pengembangan (Dev) dan Operasi (Ops) untuk memungkinkan ketangkasan dan kualitas penyampaian layanan. Istilah ini pertama kali diciptakan pada tahun 2009 dalam sebuah konferensi dan sejak saat itu telah mewujudkan siklus pengembangan sistem ujung-ke-ujung yang sering menghadirkan fitur, perbaikan, dan *update* yang sejalan dengan tujuan bisnis. Ini menekankan pada pengiriman berkelanjutan, otomatisasi, dan penerapan berkelanjutan.

Selama lima dekade terakhir pergeseran budaya dari Waterfall ke Agile dan akhirnya, ke DevOps memerintahkan motivasi dan kepemimpinan yang kuat untuk mengatasi tantangan yang terkait dengan perubahan revolusioner dalam kerangka SDLC dan membutuhkan tingkat adaptasi yang tinggi di antara tim yang terlibat dalam pengembangan perangkat lunak. . Evolusi proses membantu mengatasi kekurangan sistem yang berlaku; perubahan semacam itu terkait langsung dengan keuntungan, kesuksesan dan keuntungan

perusahaan, dan organisasi yang menolak untuk berubah hanya berisiko tertinggal dalam perlombaan untuk memberikan produk bernilai tinggi kepada pelanggan.

3. Tahapan SDLC

a. Planning/ Perencanaan Sistem

Tahap perencanaan sistem adalah proses awal untuk memahami mengapa sistem informasi harus dibangun dan menentukan bagaimana pengembang akan membangunnya. Langkah pertama yang dilakukan adalah melakukan investigasi awal. Dari hasil investigasi awal, survei diperluas menjadi studi kelayakan yang lebih rinci. Studi kelayakan adalah pengujian proposal sistem sesuai dengan kemampuan kerjanya. Dampak pada organisasi, kemampuan untuk memenuhi kebutuhan pengguna, dan penggunaan sumber daya secara efektif. Ini berfokus pada tiga pertanyaan utama:

- 1) Apa kebutuhan pengguna yang dapat dibuktikan dan bagaimana sistem kandidat memenuhinya?
- 2) Sumber daya apa yang tersedia untuk sistem kandidat tertentu? Apakah masalahnya layak dipecahkan?
- 3) Apa kemungkinan dampak dari sistem kandidat pada organisasi? Seberapa cocok itu dengan rencana induk MIS organisasi?

Setiap pertanyaan ini harus dijawab dengan hati-hati. Mereka berputar di sekitar penyelidikan dan evaluasi masalah, identifikasi dan deskripsi sistem kandidat, spesifikasi atau kinerja dan biaya setiap sistem dan pemilihan akhir dari sistem terbaik.

Tujuan dari studi kelayakan bukanlah untuk memecahkan masalah tetapi untuk mengetahui ruang lingkupnya. Selama studi definisi masalah dikristalisasi dan aspek masalah yang akan dimasukkan ke dalam sistem ditentukan. Akibatnya, biaya dan manfaat diperkirakan dengan lebih akurat pada tahap ini.

Hasil studi kelayakan berupa proposal formal. Ini hanyalah sebuah laporan dokumen formal yang merinci sifat dan ruang lingkup solusi yang diusulkan. Proposal tersebut merangkum apa saja yang diketahui dan apa

saja yang perlu dilakukan, terdiri dari:

- 1) Pernyataan masalah yang diucapkan dengan cermat yang mengarah ke analisis.
- 2) Ringkasan temuan dan rekomendasi daftar temuan utama dan rekomendasi penelitian. Ini sangat ideal untuk pengguna yang membutuhkan akses cepat ke hasil analisis sistem yang diteliti. Kesimpulan dinyatakan diikuti dengan daftar rekomendasi dan justifikasi untuk itu.
- 3) Rincian temuan garis besar metode dan prosedur yang dilakukan oleh sistem yang ada diikuti dengan cakupan tujuan dan prosedur sistem kandidat. Termasuk juga diskusi tentang laporan keluaran, struktur file, dan biaya dan manfaat dari sistem kandidat.
- 4) Rekomendasi dan kesimpulan- rekomendasi khusus mengenai sistem kandidat termasuk penugasan personel, biaya, proyek jadwal, dan tanggal target.

Setelah manajemen meninjau proposal, itu menjadi kesepakatan formal yang membuka jalan untuk desain dan implementasi yang sebenarnya. Ini adalah titik keputusan penting dalam siklus hidup. Banyak proyek mati di sini, sedangkan yang lebih menjanjikan terus berlanjut melalui implementasi. Perubahan dalam proposal dibuat secara tertulis, tergantung pada ukuran kompleksitas, dan biaya proyek. Masuk akal untuk memverifikasi perubahan sebelum melakukan desain proyek.

b. Analysis/ Analisis Sistem

Langkah berikutnya dalam proses pengembangan sistem adalah analisis sistem. Analisis sistem bertujuan untuk memungkinkan pengembang memiliki pemahaman yang lebih detail tentang masalah dan persyaratan yang memicu proyek. Oleh karena itu, domain bisnis (ruang lingkup proyek yang ditentukan selama perencanaan sistem) dapat diteliti dan dianalisis untuk memahami lebih detail apa yang berfungsi, apa yang tidak berguna, dan apa yang dibutuhkan. Analisis sistem membutuhkan kerja sama dengan pengguna sistem untuk menentukan persyaratan bisnis dan ekspektasi untuk sistem baru apa pun yang akan dibeli atau dikembangkan. Juga. prioritas bisnis mungkin perlu ditetapkan Jika jadwal dan anggaran tidak mencukupi untuk diselesaikan sesuai yang diinginkan.

Dalam fase ini, tim akan menyelidiki sistem saat ini, mengidentifikasi peluang peningkatan, dan mengembangkan konsep untuk sistem baru. Tahap ini mencakup tiga langkah:

1. Strategi analitis dibuat untuk memandu tim pengembang. Strategi ini sering melibatkan analisis sistem saat ini beserta masalahnya dan memperkenalkan pendekatan baru untuk desain sistem.
2. Langkah berikutnya adalah pengumpulan persyaratan (misalnya, melalui kuesioner ataupun wawancara). Analisa informasi dan masukan dari manajer proyek dan banyak lainnya mengarah pada pengembangan konsep sistem baru.
3. Hasil analisis, konsep dan model sistem digabungkan menjadi satu dokumen yang disebut proposal sistem, yang akan diserahkan kepada sponsor proyek dan pembuat keputusan utama lainnya (misalnya, anggota komite persetujuan), yang memutuskan apakah akan melanjutkan proyek.

Konsep dan model sistem digabungkan menjadi dokumen yang disebut proposal sistem, yang diajukan ke sponsor proyek. Proposal sistem adalah pernyataan awal yang menjelaskan persyaratan bisnis yang harus dipenuhi oleh sistem baru. Karena ini sebenarnya merupakan langkah pertama dalam merancang sistem baru, beberapa ahli berpendapat bahwa tidak tepat menggunakan istilah "analisis" sebagai nama untuk tahap ini. Beberapa orang berpikir bahwa nama yang lebih baik adalah "analisis dan desain awal". Namun, sebagian besar organisasi terus menggunakan analisis nama pada tahap ini. Harus diingat bahwa hasil dari tahap analisis adalah analisis tingkat tinggi awal dan desain sistem baru, dan keputusan besar lainnya (misalnya, anggota komite persetujuan) menentukan apakah proyek harus dilanjutkan.

c. Design/ Perancangan Sistem

Setelah memahami kebutuhan sistem informasi, kita dapat melanjutkan ke perancangan sistem. Selama perancangan sistem, perlu dicari solusi teknis alternatif terlebih dahulu. Ada beberapa solusi untuk masalah apa pun. Misalnya, sebagian besar perusahaan saat ini perlu memilih antara membeli solusi yang masuk akal dan membangun solusi

khusus mereka sendiri.

Setelah alternatif teknis dipilih dan disetujui, dalam fase ini akan dikembangkan *blueprint* teknis dan spesifikasi yang diperlukan untuk mengimplementasikan solusi akhir. *Blueprint* dan spesifikasi akan digunakan untuk mengimplementasikan database, program, antarmuka pengguna, dan jaringan yang diperlukan untuk membuat sebuah sistem.

Fase desain sistem menentukan mode operasi sistem sesuai dengan *hardware*, *software* dan infrastruktur jaringan; *user interface*, formulir dan laporan; dan prosedur khusus, database dan file yang diperlukan. Meskipun keputusan paling strategis tentang sistem dibuat selama pengembangan konsep sistem dalam tahap analisis, langkah-langkah dalam tahap desain dapat secara akurat menentukan bagaimana sistem beroperasi. Fase desain mencakup empat langkah:

- 1) Pertama, strategi dikembangkan untuk menentukan apakah sistem akan dikembangkan oleh programmer mereka sendiri, apakah sistem sedang ditransfer ke perusahaan lain (biasanya perusahaan konsultan), atau apakah perusahaan akan membeli paket *software* yang ada.
- 2) Kedua, pengembangan perancangan desain dasar sistem, yang menjelaskan *hardware*, *software* dan jaringan yang akan digunakan. Dalam beberapa kasus, sistem menambah atau mengubah infrastruktur yang ada. Desain *interface* menentukan bagaimana *user* bisa menggunakan sistem (seperti metode navigasi, menu dan tombol pada layar) serta formulir dan laporan yang akan dipakai sistem.
- 3) Ketiga, pengembangan *database* dan spesifikasi dokumen. Ini menentukan data apa yang akan disimpan dan di mana sistem akan menyimpannya.
- 4) Keempat, mengembangkan desain program yang menentukan program yang akan ditulis dan operasi yang akan dilakukan setiap program.

4. Implementation/ Implementasi Sistem

Tahap akhir dari siklus hidup sistem adalah implementasi, di mana sistem benar-benar akan dibangun dan dikembangkan. Ini biasanya merupakan tahap yang paling mengkhawatirkan, karena untuk kebanyakan sistem, ini adalah bagian tunggal yang terpanjang dan paling mahal dari proses pengembangan.

Tahap ini mencakup tiga langkah:

- a. Pertama adalah pembangunan sistem. Sistem dibangun dan diuji untuk memastikan berfungsi sebagaimana mestinya. Karena biaya kesalahan bisa jadi tinggi, pengujian merupakan salah satu langkah terpenting dalam proses implementasi. Dibandingkan dengan menulis program dari awal, sebagian besar organisasi mencurahkan lebih banyak waktu dan energi untuk pengujian. Pengujian sistem memeriksa kesiapan dan akurasi sistem untuk mengakses, memperbarui, dan mengambil data dari file baru. Setelah program tersedia, data pengujian dibaca ke komputer dan diproses terhadap file yang disediakan untuk pengujian. Jika berhasil, program akan berjalan dengan data "real-time". Jika tidak, proses diagnostik akan digunakan untuk menemukan dan memperbaiki kesalahan dalam program. Dalam kebanyakan konversi, proses paralel dilakukan dengan sistem baru yang berjalan bersamaan dengan sistem "lama". Meskipun metode ini mahal, namun dapat memberikan jaminan tambahan untuk kesalahan dalam sistem kandidat, dan juga memberikan kesempatan kepada pengguna untuk mendapatkan pengalaman melalui operasi. Namun, dalam beberapa kasus, pemrosesan paralel tidak praktis. Misalnya, tidak masuk akal untuk menjalankan dua sistem titik penjualan (POS) online paralel untuk rantai ritel. Namun, setelah sistem calon terbukti, sistem lama ditinggalkan.
- b. Kedua proses instalasi sistem. Instalasi adalah tahap di mana sistem lama diganti dengan sistem yang baru dibuat. Salah satu hal terpenting dari tahap ini adalah mengembangkan rencana pelatihan untuk memberikan pemahaman kepada *user* cara menggunakan sistem yang baru dan membantu mengelola perubahan yang terjadi. Setelah fase instalasi selesai, pengguna dapat beradaptasi dengan perubahan yang dilakukan pada sistem baru, dan evaluasi serta pemeliharaan dimulai.
- c. Ketiga tim analis membuat rencana pendukung sistem. Rencana tersebut biasanya mencakup tinjauan pasca implementasi, dan pendekatan sistematis untuk mengidentifikasi perubahan yang diperlukan oleh sistem. Seperti sistem lainnya, ada proses penuaan yang membutuhkan perawatan rutin perangkat keras dan perangkat lunak. Jika informasi baru tidak memenuhi spesifikasi desain, maka harus diubah. Perangkat keras juga membutuhkan perawatan rutin untuk memenuhi spesifikasi desain. Pentingnya pemeliharaan adalah untuk menjaga agar sistem baru tetap

sesuai standar.

5. Alat Pengembangan

Selama menjalankan metodologi pengembangan sistem dari mulai perencanaan hingga implementasi diperlukan alat dan teknik untuk melakukannya. Alat yang digunakan biasanya berupa gambar, bagan atau grafik. Selain itu digunakan pula alat non grafik, seperti kamus data, bahasa Inggris terstruktur, pseudo-code, dan formulir untuk merekam dan menyajikan data.

a. Alat-Alat Yang Berupa Grafik

Tools pengembangan sistem grafis meliputi:

1) Diagram HIPO

HIPO singkatan dari Hierarchy Plus Input Process Output, merupakan alat dokumentasi program yang basisnya fungsi, setiap modul dalam sebuah sistem dijelaskan melalui fungsi utamanya.

2) DFD

DFD singkatan dari *Data Flow Diagram*, adalah alat yang digunakan untuk menjelaskan secara terperinci sistem lama ataupun sistem baru yang akan dikembangkan secara logis tanpa melihat lingkungan fisik tempat data diekstraksi atau lingkungan fisik tempat data akan disimpan.

3) *Structured Chart*

Structured Chart merupakan alat yang digunakan untuk mendefinisikan dan menjelaskan struktur sistem informasi secara bertahap dalam bentuk modul dan sub modul dengan menampilkan hubungan antar elemen data dan elemen kontrol antar modul, sehingga dapat memberikan gambaran yang lengkap tentang sistem.

4) SADT

SADT singkatan dari *Structure Analysis and Design Technique* atau Teknik analisis dan desain terstruktur merupakan alat yang memperlakukan sistem sebagai dua bagian yakni objek dan peristiwa. Dua jenis diagram yang digunakan yaitu diagram aktivitas (*activity diagram* disebut action diagram) dan diagram data (diagram data disebut *datagram*).

5) Diagram Jackson's

Jackson's System Development yang disingkat JSD menetapkan model dunia nyata (*real world*) yang menyediakan subjek masalah sistem. Selain alat grafis yang digunakan dalam metode tertentu, ada beberapa alat grafis yang bersifat universal dan dapat digunakan di semua metode yang ada. Alat tersebut berbentuk bagan dan dapat diklasifikasikan sebagai berikut:

- a) Bagan yang menggambarkan aktivitas (*activity charting*)
 - (1) *Flowchart* sistem
 - (2) *Flowchart* program yang terdiri dari :
 - (a) *Flowchart* logika program
 - (b) *Flowchart* program computer terperinci
 - (3) *Flowchart* dokumen
 - (4) *Flowchart* proses
 - (5) Gantt chart
- b) Bagan yang menggambarkan tata letak atau *layout charting*
- c) Bagan yang menggambarkan hubungan personil atau *personil relationship charting*
 - (1) *Working distribution chart*
 - (2) *Organization chart*

b. Teknik Pengembangan Sistem

Berikut adalah teknik yang digunakan dalam pengembangan sistem :

- 1) Teknik manajemen proyek yang dipakai untuk tahap perencanaan. Antara lain CPM (*critical path method*) dan PERT (*prosedur evaluasi dan teknik review*).
- 2) Teknik pencarian fakta, yaitu teknik yang dipakai untuk mengumpulkan data dan fakta dalam kegiatan analisa sistem. Teknik tersebut antara lain meliputi:
 - a) Wawancara atau *Interview*
 - Persiapan yang dilakukan :
 - ✓ Membuat janji pertemuan
 - ✓ Memastikan orang yang akan diwawancarai
 - ✓ Menentukan pokok permasalahan
 - ✓ Perhatikan siapa yang diwawancarai
 - ✓ Mencatat tanggapan
 - ✓ Membuat jadwal kapan akan bertemu kembali

- b) Observasi atau *Observation*
 - c) Daftar pertanyaan atau *Questionnaires*
 - d) Pengumpulan Sampel atau *Sampling*
- 3) Teknik analisis biaya/ manfaat atau *Cost Effectiveness Analysis* atau *Cost Benefit Analysis* adalah Suatu teknik yang digunakan untuk menghitung biaya yang diperlukan dalam pengembangan sistem informasi.
- 4) Teknik menjalankan rapat

Tujuan diadakannya rapat dalam pengembangan sistem meliputi:

- ✓ Mendefinisikan permasalahan
- ✓ Mengumpulkan gagasan-gagasan
- ✓ Memecahkan setiap permasalahan
- ✓ Menyelesaikan setiap konflik yang terjadi
- ✓ Menganalisa perkembangan proyek
- ✓ Mengumpulkan data ataupun fakta
- ✓ Melakukan perundingan-perundingan

Tahapannya antara lain:

- ✓ Membuat Perencanaan rapat
- ✓ Menjalankan rapat sesuai rencana
- ✓ Menindaklanjuti semua hasil rapat

- 5) Teknik Inspeksi atau *Walkthrough*

Proses analisis sistem dan desain sistem perlu dipantau dan diawasi. Pengawasannya bisa dilakukan dengan melakukan verifikasi setiap hasil dari tahap pengembangan sistem yang telah selesai dilakukan. Verifikasi formal atas hasil kerja disebut inspeksi, dan secara informal disebut *walkthrough*.

C. SOAL LATIHAN/TUGAS

1. Apa yang dimaksud dengan SDLC? Jelaskan!
2. Pada tahun berapa SDLC mulai dikembangkan? Ceritakan secara singkat awal mula SDLC!
3. Dari sekian banyak metode pengembangan yang ada, mana yang menurut kalian metode yang paling baik dalam pengembangan sistem? Jelaskan!
4. Ada berapa tahapan dalam pengembangan SDLC? Jelaskan!
5. Sebutkan beberapa alat yang diperlukan dalam pengembangan sistem dan jelaskan fungsinya dalam tahapan pengembangan sistem!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Charles S. Wasson (2006). *System Analysis, Design, and Development: Concepts, Principles, and Practices*. New Jersey: John Wiley & Sons, Inc.

Jeffrey L. Whitten, Lonnie D. Bentley (2007). *Systems Analysis and Design Methods*. New York: The McGraw-Hill Companies, inc.

Shantanu Choudhary (2018). *Evolution of System Development Life Cycle (SDLC)*. Diakses dari: <https://www.linkedin.com/pulse/evolution-system-development-life-cycle-sdlc-shantanu-choudhary>

Wikipedia (2020). *Software Development Process*. Diakses dari : [https://en.wikipedia.org/wiki/Software_development_process#:~:text=The%20software%20development%20methodology%20\(also,framework%20for%20building%20information%20systems](https://en.wikipedia.org/wiki/Software_development_process#:~:text=The%20software%20development%20methodology%20(also,framework%20for%20building%20information%20systems).

GLOSARIUM

Blueprint atau Cetak Biru adalah kerangka rinci, sebagai dasar perumusan kebijakan, meliputi penetapan tujuan dan sasaran, perumusan strategi, rencana pelaksanaan dan prioritas kegiatan, serta langkah-langkah atau implementasi yang harus dilakukan setiap departemen di lingkungan kerja.

Data merupakan kumpulan angka dan karakter yang tidak berarti. Dari data tersebut dapat diolah untuk mendapatkan informasi.

Database adalah kumpulan file yang saling terkait dan membentuk struktur data. Database minimal terdiri dari file yang cukup bagi komputer untuk beroperasi dengan cara ini.

Flowchart adalah suatu diagram yang di situ terdapat simbol-simbol tertentu dan memiliki penjelasan yang berbeda. Simbol tersebut dapat mengilustrasikan tentang urutan proses secara detail.

Gate adalah Sirkuit digital yang hanya menghasilkan dalam kondisi input tertentu. Setiap fungsi logika (seperti DAN atau ATAU) dapat ditampilkan melalui gerbang yang sesuai. Fungsi logika yang lebih kompleks dapat diwujudkan dengan menghubungkan beberapa gerbang secara seri.

Internet adalah istilah umum yang digunakan untuk menunjuk jaringan kelas dunia yang terdiri dari komputer dan layanan atau sekitar 30-50 juta pengguna komputer dan lusinan sistem informasi termasuk e-mail, Gopher, FTP dan *World Wide Web*.

PERTEMUAN 3

PERENCANAAN SISTEM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian perencanaan, perlunya perencanaan dan proses perencanaan sistem. Dari pertemuan diharapkan mahasiswa mampu mendeskripsikan apa itu perencanaan sistem.

B. URAIAN MATERI

1. Pengertian Perencanaan

Tahap perencanaan sistem adalah proses dasar untuk memahami mengapa sistem informasi harus dibangun dan menentukan bagaimana tim proyek akan membangunnya. Langkah pertama yang dilakukan adalah melakukan investigasi awal. Dari hasil investigasi awal, survei diperluas menjadi studi kelayakan yang lebih rinci. Studi kelayakan adalah pengujian proposal sistem sesuai dengan kemampuan kerjanya. Dampak pada organisasi, kemampuan untuk memenuhi kebutuhan pengguna, dan penggunaan sumber daya secara efektif.

Tujuan dari studi kelayakan bukanlah untuk memecahkan masalah tetapi untuk mengetahui ruang lingkupnya. Selama studi definisi masalah dikristalisasi dan aspek masalah yang akan dimasukkan ke dalam sistem ditentukan. Akibatnya, biaya dan manfaat diperkirakan dengan lebih akurat pada tahap ini. Hasil studi kelayakan berupa proposal formal. Ini hanyalah sebuah laporan dokumen formal yang merinci sifat dan ruang lingkup solusi yang diusulkan. Proposal tersebut merangkum apa saja yang diketahui dan apa saja yang akan dilakukan, terdiri dari:

- a. Pernyataan masalah yang diucapkan dengan cermat yang mengarah ke analisis.
- b. Ringkasan temuan dan rekomendasi daftar temuan utama dan rekomendasi penelitian. Ini sangat ideal untuk pengguna yang membutuhkan akses cepat

ke hasil analisis sistem yang diteliti. Kesimpulan dinyatakan diikuti dengan daftar rekomendasi dan justifikasi untuk itu.

- c. Rincian temuan garis besar metode dan prosedur yang dilakukan oleh sistem yang ada diikuti dengan cakupan tujuan dan prosedur sistem kandidat. Termasuk juga diskusi tentang laporan keluaran, struktur file, dan biaya dan manfaat dari sistem kandidat.
- d. Rekomendasi dan kesimpulan- rekomendasi khusus mengenai sistem kandidat termasuk penugasan personel, biaya, proyek jadwal, dan tanggal target.

2. Perlunya Perencanaan

Perencanaan system merupakan Langkah awal dalam pengembangan system. Perencanaan system bertujuan agar pengembangan system sesuai dengan *blueprint* yang telah disepakati oleh pengambil kebijakan. Pengambil kebijakan dalam sebuah organisasi adalah pejabat eksekutif. Namun, terkadang perlu pendapat bawahan maupun pengguna aplikasi dalam mengambil keputusan.

Oleh karena itu, saat sebuah proyek didefinisikan kita harus mengetahui kebutuhan para pengambil keputusan. Berikut adalah orang-orang yang terlibat dalam pengambil keputusan.

a. Manajemen level atas

Prioritas utama bagi para eksekutif adalah laba atas investasi. Oleh karena itu, sistem yang akan dibangun harus dapat meningkatkan laba atas investasi.

b. Manajer level menengah

Tugas utamanya adalah meningkatkan efisiensi kerja. Oleh karena itu, sistem yang akan dibuat harus dapat menunjukkan seberapa besar peningkatan dalam hal efisiensi kerja.

c. Pengguna sistem langsung

Kebutuhan utama *user* adalah sebuah aplikasi yang dapat memudahkan pekerjaannya. Jika pengguna berpartisipasi dalam pengambilan keputusan, maka harus dapat diperlihatkan bagaimana pengguna akan mendapatkan keuntungan dari penggunaan system baru.

Pada tahap ini file yang dihasilkan adalah file proposal proyek. Dokumen

proposal proyek minimum meliputi:

- a. Keuntungan yang diperoleh pengguna dari sistem informasi yang akan dikembangkan.
- b. Biaya yang diperlukan untuk pengembangan.
- c. Waktu yang diperlukan untuk melakukan pengembangan sistem.

3. Proses Perencanaan Sistem

Selama fase permulaan Proses Terpadu dari proyek pengembangan sistem baru, seorang manajer, anggota staf, perwakilan penjualan, atau analis sistem biasanya mengidentifikasi beberapa nilai bisnis yang dapat diperoleh dari penggunaan teknologi informasi. Proyek pengembangan sistem baru harus dimulai dari kebutuhan atau peluang bisnis. Banyak ide untuk sistem baru atau peningkatan dari yang sudah ada muncul dari penerapan teknologi baru, tetapi pemahaman tentang teknologi biasanya sekunder dari pemahaman yang kuat tentang bisnis dan tujuannya. Ini tidak berarti bahwa orang teknis tidak boleh merekomendasikan proyek sistem baru. Faktanya, situasi yang ideal adalah bagi staf TI (yaitu, pakar sistem) dan staf bisnis (yaitu, pakar bisnis) untuk bekerja sama menemukan cara di mana teknologi dapat memenuhi kebutuhan bisnis. Untuk memastikan bahwa kebutuhan bisnis yang nyata sedang ditangani, organisasi bisnis yang terpengaruh (disebut sponsor proyek), mengusulkan proyek pengembangan sistem baru menggunakan permintaan sistem. Permintaan sistem secara efektif akan dimulai fase untuk proyek pengembangan sistem baru. Permintaan tersebut diteruskan ke komite persetujuan untuk dipertimbangkan. Komite persetujuan meninjau permintaan tersebut dan membuat keputusan awal apakah akan menyelidiki proposal atau tidak. Jika panitia awalnya menyetujui permintaan tersebut, tim pengembangan sistem mengumpulkan lebih banyak informasi untuk menentukan kelayakan proyek.

Analisis kelayakan memainkan peran penting dalam memutuskan apakah akan melanjutkan proyek pengembangan sistem informasi. Ini memeriksa pro dan kontra teknis, ekonomi, dan organisasi dari pengembangan sistem, dan ini memberi organisasi gambaran yang sedikit lebih rinci tentang keuntungan berinvestasi dalam sistem serta kendala yang mungkin timbul. Dalam banyak kasus, sponsor proyek bekerja sama dengan tim pengembangan untuk

mengembangkan analisis kelayakan. Setelah analisis kelayakan selesai, itu diserahkan ke komite persetujuan, bersama dengan permintaan sistem yang direvisi. Komite kemudian memutuskan apakah akan menyetujui proyek, menolak proyek, atau membuat tabel sampai informasi tambahan tersedia. Proyek dipilih dengan menimbang risiko dan keuntungan dan dengan membuat trade-off di tingkat organisasi.

Berikut adalah tahapan-tahapan yang dilakukan dalam proses perencanaan sistem :

a. Identifikasi Proyek

Sebuah proyek diidentifikasi ketika seseorang dalam organisasi mengidentifikasi kebutuhan bisnis untuk mengembangkan sistem. Hal ini dapat terjadi dalam unit bisnis atau IT, berasal dari komite pengarah yang bertugas mengidentifikasi peluang bisnis, atau berkembang dari arahan yang dibuat oleh konsultan dari luar. Contoh kebutuhan bisnis termasuk mendukung promosi pemasaran baru, menjangkau jenis *customer* baru, atau meningkatkan hubungan dengan pemasok. Terkadang, kebutuhan muncul dari semacam "rasa sakit" di dalam organisasi, seperti penurunan pangsa pasar, tingkat layanan pelanggan yang buruk, atau persaingan yang meningkat. Di lain waktu, inisiatif dan strategi bisnis baru dibuat, dan sistem diperlukan untuk memungkinkannya.

Kebutuhan bisnis bisa juga muncul saat organisasi mengidentifikasi cara unik dan masuk akal dalam menggunakan teknologi informasi. Banyak organisasi melihat perkembangan teknologi, yaitu teknologi yang masih berkembang dan belum dapat digunakan untuk keperluan bisnis secara luas. Misalnya, jika perusahaan tetap mengikuti perkembangan teknologi seperti augmented reality, game, kartu pintar, dan perangkat seluler, mereka dapat mengembangkan strategi bisnis yang memanfaatkan kemampuan teknologi ini dan memperkenalkannya ke pasar sebagai penggerak pertama. Idealnya, mereka dapat memanfaatkan keunggulan penggerak pertama ini dengan menghasilkan uang dan terus berinovasi sementara pesaing tertinggal.

Sponsor proyek adalah seseorang yang menyadari kebutuhan bisnis yang kuat akan suatu sistem dan berkepentingan untuk melihat sistem tersebut berhasil. Dia akan bekerja sepanjang expositions pengembangan untuk memastikan agar proyek bergerak sesuai arah yang benar dari sudut pandang bisnis. Support proyek bertindak sebagai titik kontak utama untuk

sistem. Biasanya support proyek berasal dari fungsi bisnis seperti pemasaran, akuntansi atau keuangan. Namun, anggota bidang TI juga dapat mensponsori atau mensponsori proyek.

Jenis sponsor yang diperlukan dapat ditentukan dari ukuran atau ruang lingkup proyek. Sistem untuk departemen skala kecil mungkin memerlukan sponsor hanya dari satu manajer, sedangkan inisiatif organisasi besar mungkin memerlukan dukungan dari seluruh tim manajemen senior dan bahkan CEO. Bila sebuah proyek murni bersifat teknis (misalkan Peningkatan infrastruktur teknologi informasi yang ada atau penelitian tentang kelangsungan teknologi yang muncul), maka sponsor dari TI sudah tepat. Ketika proyek memiliki arti penting bagi bisnis namun secara teknis rumit, sponsor bersama oleh bisnis dan TI mungkin diperlukan.

Persyaratan bisnis tingkat tinggi untuk sistem didorong oleh kebutuhan bisnis. Persyaratan adalah apa saja yang akan dikerjakan oleh sistem informasi, atau fungsi-fungsi yang dikandungnya. Hal ini perlu dijelaskan pada tingkat tinggi sehingga komite persetujuan dan, pada akhirnya, tim proyek memahami apa yang diharapkan bisnis dari produk akhir. Persyaratan bisnis adalah fitur dan kapabilitas yang harus dimiliki sistem informasi, seperti kemampuan untuk mengumpulkan pesanan pelanggan secara online atau kemampuan pemasok untuk menerima informasi inventaris saat pesanan dilakukan dan penjualan dilakukan.

Gagasan tentang nilai bisnis yang akan diperoleh dari sistem juga harus dimiliki oleh sponsor proyek, baik dengan cara yang berwujud ataupun tidak berwujud. Nilai nyata dapat diukur dan diukur dengan mudah (misalnya, pengurangan 2 persen dalam biaya operasi). Hasil nilai tak berwujud dari keyakinan yang peka bahwa sistem memberikan manfaat yang penting, tetapi sulit diukur, bagi organisasi (misalnya, layanan pelanggan yang lebih baik atau posisi kompetitif yang lebih baik).

Setelah sponsor proyek mengidentifikasi proyek yang memenuhi kebutuhan bisnis penting dan dia dapat mengidentifikasi persyaratan dan nilai bisnis sistem, sekarang saatnya untuk memulai proyek secara resmi. Di sebagian besar organisasi, awal mula proyek dimulai dengan dokumen yang dinamakan *system request* (permintaan sistem).

b. Analisis Kelayakan

Setelah kebutuhan akan sistem dan persyaratan bisnisnya telah

ditentukan, sekarang saatnya untuk membuat kasus bisnis yang lebih detail untuk lebih memahami peluang dan batasan yang terkait dengan proyek yang diusulkan. Analisis kelayakan mengarahkan organisasi untuk menetapkan apakah akan melanjutkan proyek atau tidak. Analisis kelayakan juga memeriksa risiko penting terkait proyek, yang harus diselesaikan jika proyek disetujui. Seperti persyaratan sistem lainnya, setiap organisasi memiliki proses dan format analisis kelayakannya sendiri, tetapi kebanyakan mencakup tiga jenis: kelayakan teknis, kelayakan ekonomi, dan kelayakan organisasi. Hasil analisis digabungkan menjadi studi kelayakan dan kemudian diserahkan kepada komite persetujuan.

Meskipun sekarang kita membahas analisis kelayakan dalam konteks memulai proyek, sebagian besar tim proyek akan merevisi studi kelayakan mereka selama proses pengembangan dan meninjau kembali isinya di berbagai pos pemeriksaan selama proyek berlangsung. Jika sewaktu-waktu risiko dan batasan proyek melebihi manfaatnya, tim proyek dapat menentukan untuk pembatalan proyek atau melakukan perbaikan yang diperlukan.

c. Pemilihan Proyek

Setelah analisis kelayakan selesai, itu diserahkan ke komite persetujuan, bersama dengan permintaan sistem yang direvisi. Komite kemudian memutuskan apakah akan menyetujui proyek, menolak proyek, atau membuat tabel sampai informasi tambahan tersedia. Di tingkat proyek, panitia mempertimbangkan nilai proyek dengan memeriksa kebutuhan bisnis (ditemukan dalam permintaan sistem) dan risiko membangun sistem (disajikan dalam analisis kelayakan).

Namun, sebelum menyetujui proyek, panitia juga mempertimbangkan proyek dari perspektif organisasi; itu harus mengingat seluruh portofolio proyek perusahaan. Cara mengelola proyek ini disebut manajemen portofolio. Manajemen portofolio mempertimbangkan berbagai jenis proyek yang ada dalam sebuah organisasi besar dan kecil, risiko tinggi dan risiko rendah, strategis dan taktis. Portofolio proyek yang baik memiliki campuran proyek yang paling tepat untuk kebutuhan organisasi. Komite bertindak sebagai manajer portofolio dengan tujuan memaksimalkan kinerja biaya-manfaat dan faktor penting lainnya dari proyek dalam portofolio mereka. Misalnya, sebuah organisasi mungkin ingin mempertahankan proyek berisiko

tinggi kurang dari 20 persen dari total portofolio proyeknya.

Komite persetujuan harus selektif tentang ke mana harus mengalokasikan sumber daya. Ini melibatkan pertukaran di mana organisasi harus menyerahkan sesuatu sebagai imbalan untuk sesuatu yang lain untuk menjaga portofolionya seimbang. Jika ada tiga proyek yang berpotensi memberi hasil tinggi, namun semuanya memiliki risiko sangat tinggi, maka mungkin hanya satu proyek yang akan dipilih. Juga, ada kalanya sistem di tingkat proyek masuk akal bisnis yang baik, tetapi tidak masuk akal di tingkat organisasi. Dengan demikian, sebuah proyek dapat menunjukkan ROI (*Return On Investment*) yang sangat kuat dan mendukung kebutuhan bisnis yang penting untuk sebagian perusahaan, tetapi tidak dipilih. Hal ini dapat terjadi karena berbagai alasan karena tidak ada dana dalam anggaran untuk sistem lain, organisasi akan mengalami beberapa jenis perubahan (misalnya, merger), proyek yang memenuhi persyaratan bisnis yang sama sedang berjalan, atau sistem tidak selaras dengan strategi perusahaan saat ini atau di masa depan.

d. Estimasi Penyelesaian Proyek

Ilmu atau seni manajemen proyek berbobot antara tiga konsep penting: fungsi sistem, waktu penyelesaian proyek (saat proyek selesai) dan biaya proyek. Pikirkan ketiga hal ini sebagai pengungkit yang saling bergantung yang dikontrol oleh manajer proyek selama pengembangan sistem. Setiap kali satu tuas ditarik, dua tuas lainnya akan terpengaruh entah bagaimana. Misalnya, jika manajer proyek perlu menyesuaikan tenggat waktu ke tanggal yang lebih awal, satu-satunya solusi adalah mengurangi fungsi sistem atau meningkatkan biaya dengan menambah staf atau membuat mereka bekerja lembur. Seringkali, manajer proyek harus bekerja dengan sponsor proyek untuk mengubah tujuan proyek, seperti mengembangkan sistem dengan fungsionalitas yang lebih sedikit atau memperpanjang tenggat waktu untuk sistem akhir, sehingga proyek tersebut memiliki tujuan yang wajar yang dapat dipenuhi. Pada tahap awal proyek, manajer perlu mengevaluasi setiap pengungkit, dan kemudian melanjutkan untuk mengevaluasi bagaimana memulai proyek dengan cara yang sesuai dengan kebutuhan organisasi. Estimasi adalah proses menetapkan nilai estimasi waktu dan tenaga. Perkiraan yang dibuat di awal proyek biasanya

didasarkan pada serangkaian kemungkinan nilai dan secara bertahap menjadi lebih spesifik seiring dengan kemajuan proyek. Artinya, rentang nilai untuk fase permulaan akan jauh lebih besar daripada untuk fase transisi.

Angka yang digunakan untuk menghitung perkiraan ini dapat diambil dari proyek dengan tugas dan teknologi serupa atau disediakan oleh pengembang berpengalaman. Secara umum, angkanya harus konservatif. Praktik yang baik adalah melacak nilai aktual untuk waktu dan upaya selama proses pengembangan sehingga angka dapat disaring di sepanjang jalan dan proyek berikutnya dapat memperoleh manfaat dari data nyata.

Ada banyak cara untuk memperkirakan waktu yang dibutuhkan untuk membangun sistem. Karena Proses Terpadu didorong kasus penggunaan, kami menggunakan pendekatan yang didasarkan pada kasus penggunaan: poin kasus penggunaan. Poin use case, awalnya dikembangkan oleh Gustav Karner dari Objectory AB, didasarkan pada fitur unik dari use case dan orientasi objek. Dari sudut pandang praktis, untuk memperkirakan upaya menggunakan poin use case, use case dan use case diagram harus sudah dibuat.

Model *use case* memiliki dua konstruksi utama: aktor dan use case. Aktor merepresentasikan peran yang dimainkan oleh pengguna sistem, bukan pengguna tertentu. Misalnya, peran bisa menjadi sekretaris atau manajer. Aktor juga dapat mewakili sistem lain yang akan berinteraksi dengan sistem yang sedang dikembangkan. Untuk tujuan estimasi poin kasus penggunaan, aktor dapat diklasifikasikan sebagai sederhana, rata-rata, atau kompleks. Aktor sederhana adalah sistem terpisah yang dengannya sistem saat ini harus berkomunikasi melalui antarmuka program aplikasi (API) yang didefinisikan dengan baik.

Aktor rata-rata adalah sistem terpisah yang berinteraksi dengan sistem saat ini menggunakan protokol komunikasi standar, seperti TCP / IP, FTP, atau HTTP, atau database eksternal yang dapat diakses menggunakan SQL standar. Aktor kompleks biasanya adalah pengguna akhir yang berkomunikasi dengan sistem. Setelah semua aktor dikategorikan sebagai sederhana, rata-rata, atau kompleks, manajer proyek menghitung jumlah aktor di setiap kategori dan memasukkan nilai ke dalam tabel bobot aktor yang tidak disesuaikan yang terdapat dalam lembar kerja estimasi poin *use case*. Manajer proyek kemudian menghitung Total Bobot Aktor yang Tidak

Disesuaikan. Ini dihitung dengan menjumlahkan hasil individu yang dihitung dengan mengalikan faktor pembobotan dengan jumlah aktor dari setiap jenis. Misalnya, jika kita berasumsi bahwa diagram use case memiliki nol sederhana, rata-rata nol, dan empat aktor kompleks yang berinteraksi dengan sistem yang dikembangkan.

Use case mewakili proses bisnis utama yang akan dijalankan oleh sistem yang menguntungkan para pelaku dalam beberapa cara. Bergantung pada jumlah transaksi unik yang harus ditangani kasus penggunaan, kasus penggunaan dapat dikategorikan sebagai sederhana, rata-rata, atau kompleks. Kasus penggunaan diklasifikasikan sebagai sederhana jika mendukung satu hingga tiga transaksi, sebagai rata-rata jika mendukung empat hingga tujuh transaksi, atau serumit jika mendukung lebih dari tujuh transaksi. Setelah semua kasus penggunaan berhasil dikategorikan, manajer proyek memasukkan jumlah setiap jenis kasus penggunaan ke dalam tabel pembobotan kasus penggunaan yang tidak disesuaikan yang terdapat dalam lembar kerja estimasi titik kasus penggunaan. Dengan mengalikan dengan bobot yang sesuai dan menjumlahkan hasilnya, kita mendapatkan nilai untuk total bobot kasus penggunaan yang tidak disesuaikan. Misalnya, jika kita mengasumsikan bahwa kita memiliki tiga kasus penggunaan sederhana, empat kasus penggunaan rata-rata, dan satu kasus penggunaan kompleks.

Use case mewakili proses bisnis utama yang akan dijalankan oleh sistem yang menguntungkan para pelaku dalam beberapa cara. Bergantung pada jumlah transaksi unik yang harus ditangani kasus penggunaan, *use case* dapat dikategorikan sebagai sederhana, rata-rata, atau kompleks. *Use case* diklasifikasikan sebagai sederhana jika mendukung satu hingga tiga transaksi, sebagai rata-rata jika mendukung empat hingga tujuh transaksi, atau serumit jika mendukung lebih dari tujuh transaksi. Setelah semua kasus penggunaan berhasil dikategorikan, manajer proyek memasukkan jumlah setiap jenis kasus penggunaan ke dalam tabel pembobotan kasus penggunaan yang tidak disesuaikan yang terdapat dalam lembar kerja estimasi titik kasus penggunaan. Dengan mengalikan dengan bobot yang sesuai dan menjumlahkan hasilnya, kita mendapatkan nilai untuk total bobot kasus penggunaan yang tidak disesuaikan. Misalnya, jika kita mengasumsikan bahwa kita memiliki tiga kasus penggunaan sederhana, empat kasus penggunaan rata-rata, dan satu kasus penggunaan kompleks.

Selanjutnya, manajer proyek menghitung nilai poin kasus penggunaan yang tidak disesuaikan dengan hanya menjumlahkan total bobot aktor yang tidak disesuaikan dan total bobot kasus penggunaan yang tidak disesuaikan.

Estimasi berbasis poin *use case* juga memiliki serangkaian faktor yang digunakan untuk menyesuaikan nilai poin *use case*. Dalam hal ini, ada dua set faktor: faktor kompleksitas teknis (TCF) dan faktor lingkungan (EFs). Ada tiga belas faktor teknis terpisah dan delapan faktor lingkungan yang terpisah. Tujuan dari faktor-faktor ini adalah untuk memungkinkan proyek secara keseluruhan dievaluasi untuk kompleksitas sistem yang dikembangkan dan tingkat pengalaman staf pengembangan. Jelas, jenis faktor ini dapat mempengaruhi upaya yang dibutuhkan tim untuk mengembangkan sistem. Masing-masing faktor ini diberi nilai antara 0 dan 5, 0 yang menunjukkan bahwa faktor tersebut tidak relevan dengan sistem yang sedang dipertimbangkan dan 5 yang menunjukkan bahwa faktor tersebut penting agar sistem berhasil. Nilai yang ditetapkan kemudian dikalikan dengan bobotnya masing-masing. Nilai pembobotan ini kemudian dijumlahkan untuk membuat nilai faktor teknis (TFactor) dan nilai faktor lingkungan (EFactor).

e. Membuat dan Mengelola Rencana Kerja

Setelah manajer proyek mempunyai pemahaman keseluruhan tentang fungsi dan pekerjaan proyek, dia akan membuat rencana kerja, yang berupa jadwal dinamis untuk mencatat dan memeriksa semua tugas yang perlu diselesaikan selama proyek. Rencana kerja mencantumkan setiap tugas, serta informasi penting tentang tugas, seperti kapan tugas akan diselesaikan, orang yang ditugaskan untuk melaksanakan tugas, dan hasil apa pun yang akan dihasilkan. Tingkat detail dan jumlah informasi yang ditangkap oleh rencana kerja tergantung pada kebutuhan proyek dan biasanya meningkat seiring dengan kemajuan proyek.

Tujuan keseluruhan dari sistem harus dimasukkan dalam permintaan sistem. Tugas manajer proyek adalah menentukan semua tugas yang perlu dikerjakan agar tujuan ini dapat terpenuhi. Terdengar seperti tugas yang menakutkan. Bagaimana seseorang mengetahui semua pekerjaan yang dibutuhkan untuk membuat sistem yang belum pernah dibangun?

Untuk mengidentifikasi tugas salah satu caranya adalah dengan mendapatkan list tugas yang dikembangkan dan memodifikasinya. Anda dapat menggunakan daftar tugas atau metode standar sebagai titik awal.

Seperti dijelaskan dalam Bab 1, metodologi adalah metode formal (yaitu daftar langkah dan hasil) untuk mengimplementasikan proses pengembangan sistem.

Metode yang ada dapat diadopsi oleh manajer proyek, memilih langkah-langkah dan hasil yang berlaku untuk proyek yang sedang berjalan, dan menambahkannya ke rencana kerja. Jika tidak ada metode dalam organisasi, maka dapat membeli dari konsultan atau vendor, atau bisa juga menggunakan buku seperti buku teks ini sebagai panduan. Karena sebagian besar organisasi memiliki metodologi untuk proyek, dalam membuat rencana kerja cara paling populer adalah dengan menggunakan metodologi yang ada.

Misalnya, aktivitas khas dari fase permulaan alur kerja manajemen proyek akan mencakup mengidentifikasi proyek, melakukan analisis kelayakan, memilih proyek, dan memperkirakan beban kerja. Fase permulaan alur kerja persyaratan akan mencakup penentuan teknik pengumpulan dan analisis persyaratan, mengidentifikasi persyaratan fungsional dan non-fungsional, mewawancarai pemangku kepentingan, mengembangkan dokumen visi, dan mengembangkan kasus penggunaan. Mungkin tidak ada tugas yang terkait dengan memulai operasi dan mendukung tahapan alur kerja.

f. Penempatan Staff Proyek

Penempatan staf proyek termasuk menentukan berapa banyak orang yang harus dialokasikan untuk proyek, menyesuaikan keterampilan orang dengan kebutuhan proyek, memotivasi mereka untuk mencapai tujuan proyek, dan meminimalkan konflik yang terjadi dari waktu ke waktu. Penyampaian bagian manajemen proyek ini adalah rencana kepegawaian, yang menggambarkan jumlah dan jenis orang yang akan bekerja pada proyek, struktur laporan keseluruhan, dan piagam proyek, yang menjelaskan tujuan dan aturan proyek. Namun, sebelum menjelaskan bagaimana mengembangkan rencana kepegawaian, bagaimana memotivasi karyawan dan bagaimana menangani konflik, kami menjelaskan serangkaian karakteristik tim terstruktur.

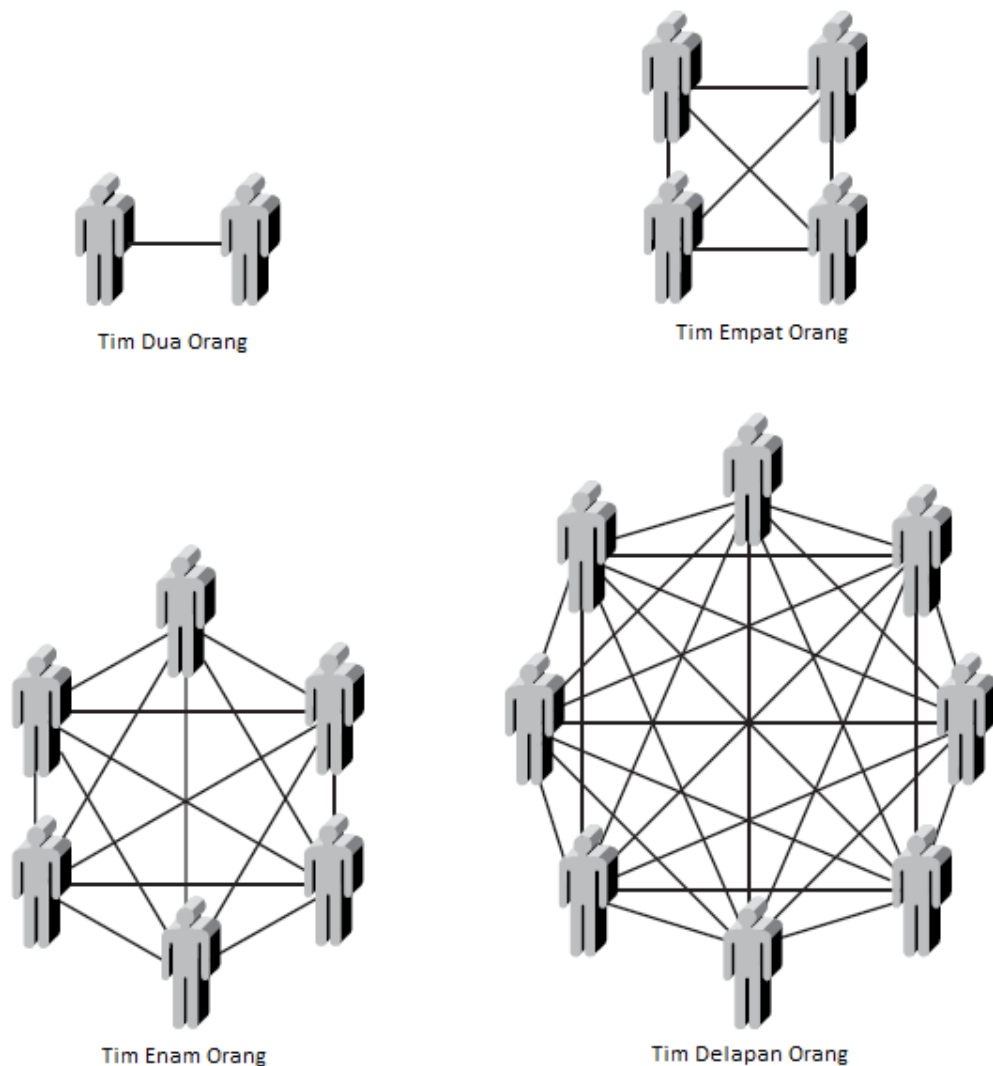
Langkah pertama dalam penyusunan staf adalah menetapkan berapa jumlah rata-rata personel yang dibutuhkan untuk proyek yang akan dijalankan. Untuk menghitung jumlah ini, bagi total beban kerja bulanan

dengan rencana terbaik. Oleh karena itu, untuk menyelesaikan proyek yang terdiri dari 40 orang per bulan dalam waktu 10 bulan, sebuah tim harus memiliki rata-rata 4 karyawan tetap, meskipun para ahli yang berbeda masuk dan keluar dari tim (seperti analis bisnis, pemrogram, dan teknisi).

Biasanya, godaannya adalah mengalokasikan lebih banyak karyawan ke suatu proyek untuk mempersingkat durasi proyek, tetapi ini bukanlah pilihan yang bijak. Meningkatkan sumber daya personel tidak berarti peningkatan produktivitas; hubungan antara jumlah karyawan dan produktivitas tidak proporsional, terutama karena lebih sulitnya mengoordinasikan sejumlah besar karyawan. Semakin banyak tim, semakin sulit pula mengelolanya. Dapat dibayangkan betapa mudahnya bekerja dalam tim proyek yang terdiri dari dua orang: anggota tim berbagi jalur komunikasi. Namun, menambahkan dua orang dapat menambah jumlah jalur komunikasi menjadi enam, dan semakin besar jumlahnya, semakin besar kompleksitas komunikasi. Gambar 3.1 mengilustrasikan dampak penambahan anggota tim ke tim proyek.

Salah satu cara untuk mengurangi hilangnya efisiensi tim adalah dengan memahami kompleksitas yang terlibat dalam angka dan menetapkan struktur pelaporan yang mengurangi dampaknya. Aturan umumnya adalah ukuran tim harus kurang dari 8 sampai 10 orang. Oleh karena itu, jika dibutuhkan lebih banyak orang, dapat dibuat sub-tim. Dengan cara ini, manajer proyek dapat memungkinkan tim kecil untuk memelihara komunikasi yang efektif, dan tim kecil dapat berkomunikasi dengan kontak tingkat yang lebih tinggi dalam proyek tersebut.

Begitu manajer proyek memahami jumlah orang yang dibutuhkan untuk proyek tersebut, dia akan mengembangkan rencana kepegawaian yang menguraikan peran yang diperlukan untuk proyek dan struktur pelaporan yang direkomendasikan. Biasanya, untuk sebuah proyek, seorang manajer proyek mengawasi kemajuan keseluruhan dari pekerjaan pengembangan, dan tim inti yang terdiri dari berbagai jenis analis dijelaskan sebelumnya. Biasanya pemimpin fungsional ditugaskan untuk mengelola sekelompok analis, sedangkan pemimpin teknis bertanggung jawab untuk mengawasi kemajuan sekelompok pemrogram dan lebih banyak staf teknis.

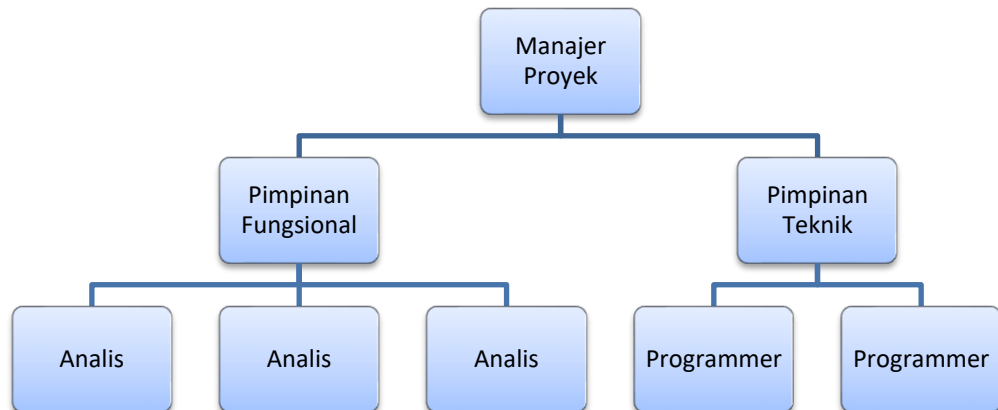


Gambar 3.1 Kompleksitas Dengan Tim Yang Lebih Besar

Ada banyak struktur untuk tim proyek; Gambar 3.2 mengilustrasikan satu kemungkinan konfigurasi tim proyek. Setelah peran ditentukan dan struktur ditentukan, manajer proyek perlu mempertimbangkan siapa yang dapat mengisi setiap peran. Biasanya, satu orang memainkan banyak peran dalam tim proyek.

Saat tugas dibuat, ingatlah bahwa orang mempunyai kemampuan teknis dan interpersonal, keduanya penting dalam sebuah proyek. Kemampuan teknis sangat berguna saat menangani tugas-tugas teknis dan mencoba memahami berbagai tugas yang dimainkan teknologi dalam proyek tertentu (misalnya, cara mengonfigurasi server web berdasarkan jumlah klik yang diharapkan pelanggan). Keterampilan interpersonal, di sisi lain,

mencakup kemampuan interpersonal dan komunikasi yang digunakan ketika berhadapan dengan pengguna bisnis, eksekutif manajemen senior, dan anggota lain dari tim proyek. Mereka sangat penting saat melakukan aktivitas pengumpulan persyaratan dan saat menangani masalah kelayakan organisasi. Setiap proyek membutuhkan keterampilan teknis dan interpersonal yang unik.



Gambar 3.2 Struktur Pelaporan Yang Memungkinkan

g. Pengelolaan Lingkungan dan Infrastruktur

Alur kerja pengelolaan lingkungan dan infrastruktur mendukung tim pengembangan selama proses pengembangan. Alur kerja lingkungan terutama berkaitan dengan pemilihan perangkat yang benar yang akan digunakan selama proses pengembangan dan mengidentifikasi rangkaian standar yang sesuai untuk diikuti selama proses pengembangan.

Alur kerja manajemen infrastruktur berkaitan dengan pemilihan tingkat dan jenis dokumentasi yang sesuai yang akan dibuat selama proses pengembangan. Aktivitas lain yang terkait dengan alur kerja manajemen infrastruktur termasuk mengembangkan, memodifikasi, dan menggunakan kembali komponen, kerangka kerja, pustaka, dan pola yang telah ditentukan sebelumnya.

C. SOAL LATIHAN/TUGAS

1. Jelaskan cara proyek pengembangan sistem diidentifikasi dan dimulai!
2. Jelaskan mengapa penting untuk menghubungkan sistem informasi dengan kebutuhan bisnis organisasi!
3. Jelaskan bagaimana proyek dipilih!
4. Buat struktur rincian kerja evolusioner!
5. Jelaskan masalah yang berkaitan dengan memotivasi pengembang perangkat lunak!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Charles S. Wasson (2006). *System Analysis, Design, and Development: Concepts, Principles, and Practices*. New Jersey: John Wiley & Sons, Inc.

GLOSARIUM

Feasibility adalah suatu kegiatan yang mengevaluasi sejauh mana manfaat dapat diperoleh dari pelaksanaan kegiatan pengembangan sistem.

Evolusioner adalah perubahan yang berangsur-angsur atau secara bertahap.

FTP (File Transfer Protocol) adalah protokol yang disediakan di Internet digunakan untuk mentransfer file dari lokasi (situs) di Internet ke komputer lokal.

HTTP (Hyper Text Transfer Protocol) adalah metode atau protokol yang digunakan untuk mengunduh file ke komputer. Protokol ini didasarkan pada hypertext, yang merupakan format teks yang umum digunakan di Internet.

Java adalah bahasa pemrograman yang digunakan untuk membuat konten aktif di halaman web, yang juga dapat dijalankan di komputer mana pun. Applet Java dapat diambil / diakses dari halaman web dan dapat dijalankan sepenuhnya.

SQL (Structured Query Language) adalah bahasa khusus domain yang digunakan dalam pemrograman untuk mengelola data yang disimpan dalam sistem manajemen basis data relasional (RDBMS).

TCP/ IP (Transmission Control Protocol/ Internet Protocol) adalah protokol komunikasi yang awalnya dikembangkan oleh Departemen Pertahanan AS. TCP / IP menyediakan jalur transmisi data sehingga data tertentu yang dikirim oleh server dapat diterima oleh server lain. TCP / IP adalah protokol yang memungkinkan sistem di seluruh dunia untuk berkomunikasi dalam satu jaringan yang disebut Internet.

Use case adalah teknik pemodelan yang digunakan untuk menjelaskan apa yang harus dilakukan oleh sistem baru. Model kasus penggunaan dibangun melalui proses berulang berdasarkan persyaratan spesifik yang disetujui semua orang selama proses diskusi antara pengembang sistem dan pelanggan (dan / atau pengguna akhir).

PERTEMUAN 4

ANALISA SISTEM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian dan proses analisa sistem. Dari pertemuan ini diharapkan mahasiswa mampu mendeskripsikan apa itu analisa sistem.

B. URAIAN MATERI

1. Pengertian Analisa Sistem

Analisis sistem merupakan kegiatan melihat sistem yang sudah ada, menganalisa bagian mana yang baik dan kurang baik, lalu mencatat segala kebutuhan yang hendak dipenuhi pada sistem yang akan dikembangkan. Terlihat seperti sederhana, padahal tidak. Banyak kendala yang akan dihadapi dalam proses ini.

Dalam beberapa proyek pengembangan sistem informasi, proses analisis dan desain biasanya dilakukan secara paralel. Oleh karena itu, selama kegiatan analisis, kegiatan perancangan juga akan dilakukan. Hal tersebut dilakukan karena dalam banyak kasus, pengguna seringkali kesulitan untuk mendefinisikan kebutuhannya. Oleh karena itu, jika melihat desain sistem yang baru, terutama desain antarmuka, akan lebih mudah bagi mereka untuk menentukan kebutuhan. Oleh karena itu, biasanya terdapat banyak perbedaan mengenai bagian mana yang dianggap analisis dan yang dianggap desain. Misalnya, beberapa orang mengatakan bahwa use case, class analysis, dan sequence diagram adalah bagian dari analisis. Namun, ada orang lain yang mengklaim bahwa use case dan sequence diagram adalah bagian dari desain, dan karena sudah ada kelas desain, tidak ada kelas analisis.

Analisis adalah inti dari proses tersebut. Ini adalah komponen kunci dari dua fase pertama siklus. Dalam analisis sistem saat ini, analis mengumpulkan sejumlah besar data yang relatif tidak terstruktur melalui wawancara, survei

kuesioner, observasi lapangan, manual proses, dll. Metode tradisional adalah untuk mengatur dan mengubah data melalui diagram alur sistem untuk mendukung pengembangan sistem di masa depan dan menyederhanakan komunikasi dengan pengguna. Tetapi diagram alir sistem mewakili sistem fisik, bukan sistem logis. Hal ini membuat sulit untuk membedakan apa yang terjadi di sistem dan bagaimana itu terjadi.

Ada masalah lain dengan pendekatan tradisional.

- a. Siklus hidup sistem memberikan kontrol kualitas yang sangat sedikit untuk memastikan komunikasi yang akurat dari pengguna ke analis. Mereka tidak memiliki bahasa yang sama.
- b. Analis dengan cepat kewalahan dengan detail bisnis dan teknis dari sistem. Sebagian besar waktu dihabiskan untuk mengumpulkan informasi. Detail diperlukan dan harus tersedia, tetapi analis tidak memiliki alat untuk menyusun dan mengontrol detail.
- c. Alat analisis yang ada memiliki keterbatasan.
 - 1) Deskripsi naratif bahasa Inggris dari suatu sistem seringkali terlalu kabur dan menyulitkan pengguna untuk memahami bagaimana bagian-bagian tersebut cocok satu sama lain. Selain itu, bahasa Inggris secara inheren sulit digunakan jika membutuhkan ketelitian.
 - 2) Bagan alur sistem dan program berkomitmen untuk implementasi fisik sistem sebelum memiliki pemahaman lengkap tentang persyaratan logisnya.
- d. Masalah juga terkait dengan spesifikasi sistem:
 - 1) Sebuah. Spesifikasi sistem sulit untuk dipelihara atau dimodifikasi. Perubahan sederhana dalam persyaratan pengguna memerlukan perubahan di beberapa bagian dokumen.
 - 2) Mereka menggambarkan kebutuhan pengguna dalam istilah perangkat keras fisik yang akan mengimplementasikan sistem daripada apa yang pengguna ingin sistem lakukan.
 - 3) Mereka monolitik dan berlebihan; Artinya, untuk mengetahui informasi tentang bagian tertentu dari sistem, pengguna harus mencari seluruh dokumen. Lebih lanjut, informasi yang sama ditemukan di banyak lokasi tanpa referensi silang.

Karena kekurangan tersebut, analis membutuhkan cetak biru yang mirip

dengan arsitek sebagai titik awal untuk desain sistem. Ini adalah metode yang berfokus pada fungsi daripada realisasi fisik. Salah satu alat tersebut adalah Data Flow Diagram (DFD).

Ada alat lainnya. Beberapa alat digunakan dalam analisis terstruktur, termasuk:

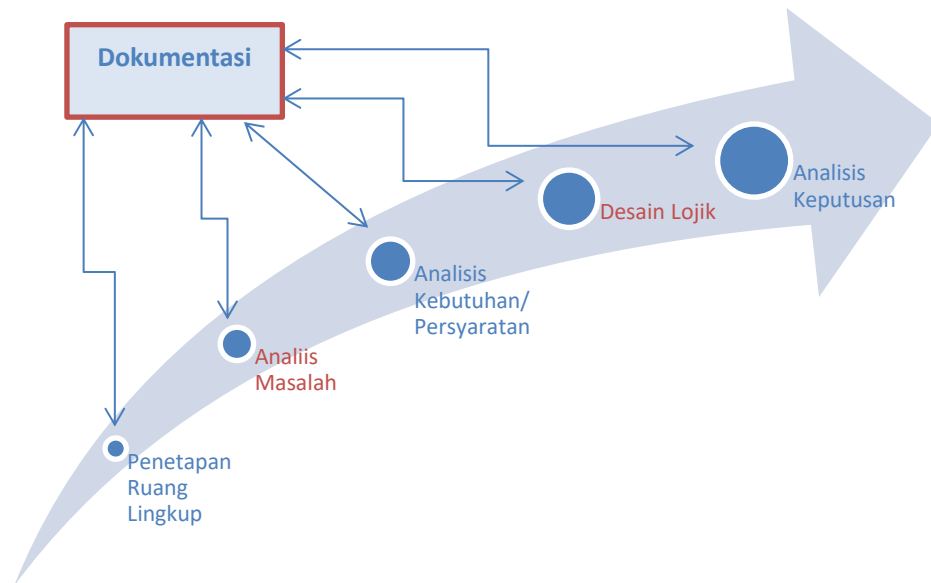
- a. DFD (*Data Flow Diagram*).
- b. Kamus Data.
- c. Bahasa Inggris Terstruktur.
- d. Pohon Keputusan.
- e. Tabel Keputusan.

Analisis sistem adalah tentang memahami situasi, bukan memecahkan masalah. Oleh karena itu, analisis yang efektif menekankan penyelidikan dan pertanyaan untuk mempelajari bagaimana sistem saat ini beroperasi dan untuk mengidentifikasi persyaratan yang dimiliki pengguna untuk yang baru atau yang dimodifikasi. Hanya setelah analisis memahami sepenuhnya sistem barulah mereka dapat menganalisisnya dan mengumpulkan rekomendasi untuk desain sistem. Cara penyelidikan sistem dilakukan akan menentukan apakah informasi yang tepat dikumpulkan. Pada gilirannya, memiliki informasi yang benar mempengaruhi kualitas aplikasi yang mengikutinya. Dengan kata lain, desain sistem yang baik, baik dikembangkan melalui metode SDLC, prototyping, atau metode terstruktur, dimulai dengan mendokumentasikan sistem saat ini dan mendiagnosis persyaratan sistem dengan benar.

2. Proses Analisa Sistem

a. Tahapan Analisa Sistem

Masalah bisnis pemilik sistem dan pengguna sistem menjadi acuan dalam analisa sistem. Dalam hal ini peran analisa sistem merupakan penghubung antara pemilik sistem dan *user*. Tahap-tahap analisa sistem dapat kita lihat pada gambar berikut ini:



Gambar 4.1 Tahapan Analisa Sistem

b. Penetapan Ruang Lingkup

Selama fase penetapan ruang lingkup, beberapa tugas harus diselesaikan, antara lain:

- 1) Identifikasi masalah awal pada sistem yang masih berjalan.
- 2) Negosiasikan cakupan proyek pengembangan sistem.
- 3) Evaluasi kelayakan proyek, seperti yang dapat dilihat pada tabel berikut:

Pernyataan Singkat Masalah	Urgensi	Visibilitas	Keuntungan	Prioritas	Solusi Yang Diusulkan
1. Waktu respon pesanan diukur dari saat menerima pesanan sampai pengiriman pelanggan meningkat rata-rata 15 per hari	Segera	Tinggi	\$ 200.000	2	Pengembangan baru
2. Ketidak konsistenan data dalam file-file anggota dan pesanan	3 bulan	Tinggi	\$ 50.000	1	Perbaikan cepat, kemudian pengembangan baru

Tabel 4.1 Penetapan Ruang Lingkup

- 4) Kembangkan jadwal dan anggaran awal.
- 5) Komunikasikan rencana proyek.

c. Analisa Masalah

Selalu ada sistem yang sudah ada atau sudah berjalan, dan tahap ini memberikan pemahaman yang lebih dalam, peluang atau perintah untuk memicu proyek untuk dianalisis. Pada tahap ini tugas yang harus diselesaikan antara lain:

- 1) Pahami bidang masalahnya. Tim analis mempelajari sistem yang sudah ada saat ini. Pemilik dan *user* sistem mempunyai pandangan berbeda tentang sistem yang sudah ada, dan penelitian yang baik adalah dapat mengungkap kebutuhan dari semua pihak.
- 2) Analisa masalah dan peluang. Meskipun telah diselesaikan pada tahap sebelumnya, masalah awal ini hanyalah gejala, bukan masalah yang bisa dimengerti oleh *user*. Analisis masalah merupakan keahlian yang sulit untuk dikuasai, sebab dan akibat dari setiap masalah akan dianalisis.
- 3) Analisa proses bisnis. Dikenal pula dengan desain ulang proses bisnis. Tim akan memeriksa semua proses bisnis secara terperinci untuk menentukan apa yang perlu ditambah atau dikurangi.
- 4) Tentukan tujuan perbaikan sistem. Tim menentukan kriteria sebagai tolak ukur dalam perbaikan system dan mengidentifikasi sejauh mana perbaikan bisa dilakukan. Kriteria bisa diukur dengan tujuan, dan setiap tujuan mewakili upaya yang harus dilakukan. Berikut ini adalah contoh analisis penentuan tujuan perbaikan sistem:

Analisa Sebab Akibat		Tujuan Perbaikan Sistem	
Masalah atau Kesempatan	Sebab dan Akibat	Tujuan Sistem	Batasan Sistem
Waktu respon pesanan terlalu lama	Sistem terlalu bergantung pada keyboard. Nilai yang sama ditunjukan pada semua pesan.	Entry data lewat keyboard berkurang 50% pada setiap pesanan	Beberapa sistem yang dikembangkan harus cocok dengan sistem windows 7 atau lebih tinggi

Tabel 4.2 Analisa Masalah

- 5) Perbarui rencana proyek.
- 6) Bertukar temuan dan saran.

d. Analisis Kebutuhan/ Persyaratan

Setelah tahap analisis masalah, hal buruknya adalah mulai mencari solusi alternatif, terutama solusi teknis. Pernyataan tersebut menunjukkan salah satu kesalahan paling umum dalam sistem informasi terbaru untuk memastikan bahwa sistem berfungsi dengan baik dan secara teknis mengesankan. Ini harus tentang "apa" daripada "bagaimana". Yang harus diperhatikan adalah apa yang pengguna benar-benar dapatkan dari sistem baru. Itu selalu dievaluasi apakah sistem baru memenuhi tujuan dan persyaratan bisnis, jadi tahap ini tidak dapat diabaikan.

Pada tahap ini, beberapa tugas harus diselesaikan, antara lain:

- 1) Mengidentifikasi dan menyebutkan persyaratan / kebutuhan sistem.
Dalam hal ini menerjemahkan tujuan menjadi persyaratan fungsional. Persyaratan fungsional adalah penjabaran kegiatan dan layanan yang harus dimiliki oleh sistem.
- 2) Memberikan prioritas kepada kebutuhan sistem. Semua persyaratan tidak sama, karena tingkat persyaratan berbeda, sehingga pemilik maupun *user* sistem harus memprioritaskan persyaratan.
- 3) Memperbarui atau meningkatkan rencana kerja. Cakupan adalah tujuan yang terus berubah. Setelah menentukan kebutuhan bisnis, kita harus mundur selangkah, membangun kembali pemahaman kita tentang ruang lingkup proyek, dan memperbarui rencana proyek kita untuk membuat penyesuaian.
- 4) Mengkomunikasikan pernyataan persyaratan atau kebutuhan.
Komunikasi merupakan tugas fase analisis kebutuhan yang dilakukan terus menerus. Kita harus mengkomunikasikan kebutuhan dan prioritas kepada dunia usaha melalui tahap ini.

e. Desain Logik

Pada tahap ini, dijelaskan berbagai model sistem untuk mendokumentasikan persyaratan sistem baru dan yang ditingkatkan.

f. Analisa Keputusan

Mengingat adanya persyaratan bisnis, maka dapat ditekankan bagaimana menggunakan teknologi untuk mengimplementasikan sistem baru. Pada tahap ini, dapat diidentifikasi solusi kandidat, menganalisis solusi

kandidat, dan merekomendasikan sistem yang akan dirancang, dibangun, dan diimplementasikan. Contoh analisis keputusan ditunjukkan pada tabel berikut:

Karakteristik	Kandidat 1	Kandidat 2	Kandidat3	Kandidat
Perangkat lunak yang diperlukan untuk mendesain dan membangun kandidat solusi	MS Visual C++ dan MS Access	MS Visual Basic 5.0, System Architect 3.1 dan Internet Explore	MS Visual Basic 7.0, System Architect 4.1 dan Internet explorer	MS Visual Studio dan MS Edge

Tabel 4.3 Analisa Keputusan

3. Jenis Kebutuhan Sistem

Kebutuhan hanyalah pernyataan tentang apa yang harus dilakukan sistem atau karakteristik apa yang harus dimilikinya. Dalam proses analisis, persyaratan ditulis dari perspektif personel bisnis dan fokus pada "konten" sistem. Karena fokusnya adalah pada kebutuhan pengguna, mereka sering disebut sebagai persyaratan bisnis (terkadang juga disebut persyaratan pengguna). Kemudian dalam desain, kembangkan persyaratan bisnis untuk meningkatkan tingkat teknis, dan jelaskan cara mengimplementasikan sistem. Persyaratan desain ditulis dari perspektif pengembang dan biasanya disebut persyaratan sistem.

Hal yang perlu ditekankan di sini adalah bahwa tidak ada garis hitam dan putih yang memisahkan persyaratan bisnis dari persyaratan sistem, dan beberapa perusahaan dapat menggunakan istilah ini secara bergantian. Hal penting untuk diingat adalah bahwa persyaratan adalah persyaratan untuk fungsi sistem. Seiring dengan kemajuan proyek dari awal ke desain rinci dan kemudian ke konstruksi, persyaratan akan berubah seiring waktu. Kebutuhan berkembang dari uraian terperinci tentang fungsi bisnis yang harus dimiliki sistem hingga uraian terperinci tentang bagaimana menerapkan fungsi teknis dalam sistem baru.

Kebutuhan bisa fungsional atau non-fungsional. Persyaratan fungsional berhubungan langsung dengan proses yang harus dilakukan sistem atau

informasi yang dikandungnya. Misalnya, persyaratan fungsional adalah bahwa sistem harus dapat menemukan persediaan yang tersedia dan melaporkan biaya aktual dan yang dianggarkan. Persyaratan fungsional secara langsung berkaitan dengan penciptaan model fungsional, struktural dan perilaku yang mewakili berfungsinya sistem yang berkembang. Persyaratan non-fungsional mengacu pada karakteristik operasional yang dibutuhkan oleh sistem, seperti kinerja dan ketersediaan. Kemampuan untuk terhubung ke sistem menggunakan browser web tidak dianggap sebagai persyaratan fungsional. Persyaratan non-fungsional dapat mempengaruhi analisis lain (fungsi, struktur, model perilaku), tetapi biasanya hanya secara tidak langsung. Persyaratan non-fungsional terutama digunakan dalam desain pengambilan keputusan mengenai database, antarmuka pengguna, perangkat keras dan perangkat lunak, dan arsitektur fisik yang mendasari sistem.

Kebutuhan non fungsional menggambarkan berbagai karakteristik mengenai sistem: operasional, kinerja, keamanan, dan budaya dan politik. Persyaratan operasional menangani masalah yang terkait dengan persyaratan fisik dan teknis di mana sistem akan beroperasi. Kebutuhan kinerja mengatasi masalah yang terkait dengan kecepatan, kapasitas, dan keandalan sistem. Kebutuhan keamanan menangani masalah yang berkaitan dengan siapa yang memiliki akses ke sistem dan dalam keadaan spesifik apa. Kebutuhan budaya dan politik menangani masalah yang terkait dengan budaya, faktor politik, dan kebutuhan hukum yang memengaruhi sistem. Ciri-ciri ini tidak menggambarkan proses bisnis atau informasi, tetapi sangat penting dalam memahami seperti apa sistem akhirnya. Kebutuhan non fungsional terutama memengaruhi keputusan yang akan dibuat selama desain sistem.

Salah satu bidang pengembangan sistem informasi yang berfokus pada perbedaan kebutuhan fungsional dan non fungsional adalah kualitas perangkat lunak. Ada banyak model berbeda yang diusulkan untuk mengukur kualitas perangkat lunak. Namun, hampir semuanya membedakan kebutuhan fungsional dan nonfungsional. Dari perspektif kualitas, kualitas fungsional berkaitan dengan sejauh mana perangkat lunak memenuhi persyaratan fungsional, yaitu seberapa banyak masalah yang sebenarnya diselesaikan oleh solusi perangkat lunak yang disediakan. Padahal, persyaratan nonfungsional dikaitkan dengan dimensi efisiensi, rawatan, portabilitas, reliabilitas, usabilitas, testabilitas, dan kualitas kegunaan. Seperti yang dinyatakan di atas, dimensi

terkait non fungsional dikaitkan terutama dengan desain rinci aktual dan implementasi sistem. Dimensi non fungsional eksternal meliputi efisiensi, reliabilitas, dan kegunaan, sedangkan dimensi nonfungsional internal meliputi rawatan, portabilitas, usabilitas ulang, dan kemampuan pengujian.

Dari perspektif pengguna, dimensi eksternal lebih penting. Jika sistem terlalu sulit untuk digunakan, terlepas dari seberapa baik sistem menyelesaikan masalah, pengguna tidak akan menggunakan sistem. Dengan kata lain, dari perspektif pengguna, agar sistem informasi berhasil, sistem tersebut tidak hanya harus memenuhi spesifikasi fungsional, tetapi juga harus memenuhi spesifikasi nonfungsional eksternal. Dari perspektif pengembang, dimensi internal juga penting. Misalnya, mengingat bahwa sistem yang berhasil cenderung berumur panjang dan multiplatform, dimensi pemeliharaan dan portabilitas dapat memiliki implikasi strategis untuk sistem yang dikembangkan. Juga, mengingat pendekatan pengembangan tangkas yang digunakan dalam industri saat ini, pengembangan perangkat lunak yang dapat digunakan kembali dan dapat diuji sangat penting.

4. Teknik Penghimpunan Data

Seorang analis mirip dengan seorang detektif (dan pengguna sistem terkadang seperti tersangka yang sulit dimengerti). Analis tahu ada masalah yang harus dipecahkan, jadi dia perlu mencari petunjuk untuk memperjelas solusinya. Sayangnya, petunjuknya tidak selalu jelas, jadi analis perlu memperhatikan detail, berbicara dengan saksi, dan mengikuti petunjuk. Analis terbaik menggunakan berbagai teknik untuk meneliti persyaratan dan memastikan pemahaman tentang proses bisnis yang ada dan persyaratan sistem baru sebelum melanjutkan dengan desain. Analis tidak ingin mengetahui nanti bahwa mereka memiliki persyaratan utama yang salah kejutan seperti itu di akhir proses pengembangan dapat menyebabkan semua jenis masalah.

Proses pengumpulan persyaratan digunakan untuk membangun dukungan politik untuk sebuah proyek dan untuk membangun kepercayaan dan hubungan antara tim proyek yang membangun sistem dan pengguna akhir yang memilih apakah akan menggunakan sistem. Melibatkan seseorang dalam proses berarti bahwa tim proyek menganggap orang itu sebagai sumber daya yang penting dan menghormati pendapat mereka. Semua pemangku

kepentingan utama (mereka yang mungkin atau terpengaruh oleh sistem) harus dimasukkan dalam proses pengumpulan persyaratan. Pemangku kepentingan termasuk manajemen, karyawan, karyawan, dan bahkan beberapa pelanggan dan pemasok. Tanpa keterlibatan orang-orang kunci, mereka dapat merasa tidak dihargai dan dapat menyebabkan masalah dalam proses implementasi (misalnya, bagaimana mereka dapat mengembangkan sistem tanpa masukan).

Tantangan kedua dari pengumpulan data adalah memilih cara informasi dikumpulkan. Ada banyak teknik untuk mengumpulkan persyaratan yang bervariasi dari mengajukan pertanyaan kepada orang-orang hingga mengamati mereka bekerja. Pada bagian ini, kami berfokus pada lima teknik yang paling umum digunakan adalah wawancara, kuesioner, dan observasi. Setiap teknik memiliki kekuatan dan kelemahannya sendiri, banyak di antaranya yang saling melengkapi, sehingga sebagian besar proyek menggunakan kombinasi teknik.

a. Wawancara

Wawancara adalah teknik yang umum dilakukan dalam pengumpulan data. Wajar jika perlu mengetahui sesuatu, biasanya Anda bertanya kepada seseorang. Secara umum, wawancara dilakukan satu lawan satu, tetapi terkadang karena keterbatasan waktu wawancara dilakukan kepada beberapa orang dalam waktu yang bersamaan. Ada lima langkah dasar untuk proses wawancara antara lain adalah:

- 1) Memilih narasumber
- 2) Mempersiapkan pertanyaan untuk wawancara
- 3) Mempersiapkan keperluan wawancara
- 4) Melakukan proses wawancara
- 5) Tindak lanjut setelah melakukan wawancara.

Langkah awal dalam melakukan wawancara yakni membuat daftar jadwal wawancara siapa yang akan diwawancarai, kapan, dan untuk tujuan apa dapat dilihat contoh tabel dibawah. Orang-orang yang ada dalam jadwal wawancara ditentukan sesuai kebutuhan informasi yang diperlukan dalam proses analisis. Orang-orang ini dicantumkan pada jadwal wawancara sesuai dengan urutan wawancara.

Nama	Posisi	Tujuan Wawancara	Jadwal Pertemuan
Andri	Direktur Accounting	Visi strategis untuk sistem accounting baru	Senin, 16 November 2020 8.00-10.00
Jenifer	Manajer Bagian Pinjaman	Masalah saat ini dengan proses pinjaman	Senin, 16 November 2020 14.00-16.00
Anna Anshor	Supervisor Bagian Input Data	Prose pinjaman dan pembayaran	Rabu, 18 November 2020 10.00-12.00

Tabel 4.4 Jadwal Wawancara

Sangat umum jika daftar orang yang diwawancarai bertambah, sering kali 50 hingga 75 persen. Saat orang-orang diwawancarai, lebih banyak informasi yang dibutuhkan dan tambahan orang yang dapat memberikan informasi mungkin akan diidentifikasi. Ada tiga jenis pertanyaan dalam wawancara antara lain:

1) Pertanyaan tertutup

Pertanyaan tertutup adalah pertanyaan yang membutuhkan jawaban spesifik. Mereka mirip dengan pertanyaan pilihan ganda atau aritmatika dalam ujian. Pertanyaan tertutup digunakan ketika seorang analis mencari informasi spesifik dan tepat (misalnya, berapa banyak permintaan kartu kredit yang diterima per hari). Secara umum, pertanyaan yang tepat adalah yang terbaik. Misalnya, daripada bertanya, Apakah Anda menangani banyak permintaan? lebih baik bertanya, Berapa banyak permintaan yang Anda proses per hari? Pertanyaan tertutup memungkinkan analis untuk mengontrol wawancara dan memperoleh informasi yang mereka butuhkan. Namun, jenis pertanyaan ini tidak mengungkap mengapa jawabannya seperti itu, juga tidak mengungkap informasi yang tidak terpikirkan oleh pewawancara untuk ditanyakan sebelumnya.

2) Pertanyaan terbuka

Pertanyaan terbuka adalah pertanyaan yang menyisakan ruang untuk penjelasan dari pihak yang diwawancarai. Mereka mirip dalam banyak

hal dengan pertanyaan esai yang mungkin Anda temukan dalam ujian. Pertanyaan terbuka dirancang untuk mengumpulkan informasi yang kaya dan memberi orang yang diwawancarai lebih banyak kendali atas informasi yang diungkapkan selama wawancara. Terkadang informasi yang dipilih oleh orang yang diwawancarai untuk didiskusikan mengungkap informasi yang sama pentingnya dengan jawaban (misalnya, jika orang yang diwawancara hanya berbicara tentang departemen lain ketika ditanya tentang masalah, itu mungkin menunjukkan bahwa dia enggan untuk mengakuinya sendiri. masalah).

3) Pertanyaan penyelidikan

Pertanyaan penyelidikan menindaklanjuti apa yang baru saja dibahas untuk mempelajari lebih lanjut, dan pertanyaan tersebut sering digunakan ketika pewawancara tidak jelas tentang jawaban orang yang diwawancarai. Mereka mendorong orang yang diwawancarai untuk memperluas atau untuk mengkonfirmasi informasi dari tanggapan sebelumnya, dan mereka menandakan bahwa pewawancara mendengarkan dan tertarik dengan topik yang sedang didiskusikan. Banyak analis pemula yang enggan menggunakan pertanyaan menyelidik karena mereka takut orang yang diwawancarai akan dianggap tertantang atau karena mereka yakin hal itu menunjukkan bahwa mereka tidak memahami apa yang dikatakan orang yang diwawancarai. Jika dilakukan dengan sopan, pertanyaan menyelidik bisa menjadi alat yang ampuh dalam pengumpulan persyaratan.

Secara umum, pewawancara sebaiknya tidak menanyakan pertanyaan tentang informasi yang tersedia dari sumber lain. Misalnya, daripada menanyakan informasi apa yang digunakan untuk melakukan suatu tugas, lebih mudah untuk menunjukkan kepada orang yang diwawancarai formulir atau laporan (lihat bagian tentang analisis dokumen) dan tanyakan informasi apa yang digunakan. Hal ini membantu orang yang diwawancarai untuk fokus pada tugas dan menghemat waktu, karena orang yang diwawancara tidak perlu menjelaskan detail informasi yang dia butuhkan hanya untuk menunjukkannya di formulir atau laporan.

b. Kuesioner

Kuesioner adalah sekumpulan pertanyaan tertulis yang digunakan untuk memperoleh informasi dari individu. Kuesioner sering digunakan ketika ada banyak orang yang membutuhkan informasi dan pendapat. Dalam pengalaman kami, kuesioner adalah teknik umum dengan sistem yang ditujukan untuk digunakan di luar organisasi (misalnya, oleh pelanggan atau vendor) atau untuk sistem dengan pengguna bisnis yang tersebar di banyak lokasi geografis.

Kebanyakan orang secara otomatis memikirkan kertas ketika mereka memikirkan kuesioner, tetapi saat ini lebih banyak kuesioner yang didistribusikan dalam bentuk elektronik, baik melalui email atau di Web. Distribusi elektronik dapat menghemat banyak uang dibandingkan dengan menyebarkan kuesioner. Proses yang baik untuk digunakan saat menggunakan kuesioner mengikuti empat langkah.

Karena informasi pada kuesioner tidak dapat segera diklarifikasi untuk responden yang bingung, mengembangkan pertanyaan yang baik sangat penting untuk kuesioner. Pertanyaan dalam kuesioner harus ditulis dengan sangat jelas dan meninggalkan sedikit ruang untuk kesalahpahaman, sehingga pertanyaan tertutup cenderung paling umum digunakan. Pertanyaan harus secara jelas memungkinkan analisis untuk memisahkan fakta dari opini. Pertanyaan opini sering menanyakan responden sejauh mana mereka setuju atau tidak (misalnya, Apakah masalah jaringan umum?), Sedangkan pertanyaan faktual mencari nilai yang lebih tepat (misalnya, Seberapa sering masalah jaringan terjadi: sekali dalam satu jam, sekali sehari, sekali seminggu?). Berikut adalah pedoman desain pembuatan kuesioner :

- 1) Mulailah dengan pertanyaan yang tidak mengancam dan menarik.
- 2) Kelompokkan item menjadi beberapa bagian yang koheren secara logis.
- 3) Jangan meletakkan item penting di akhir kuesioner.
- 4) Jangan memenuhi halaman dengan terlalu banyak item.
- 5) Hindari singkatan.
- 6) Hindari item atau istilah yang bias atau menjerus.
- 7) Nomor pertanyaan untuk menghindari kebingungan.
- 8) Lakukan uji coba kuesioner untuk mengidentifikasi pertanyaan yang

mbingungkan.

9) Berikan anonimitas kepada responden.

Mungkin masalah yang paling jelas tetapi yang kadang terlewatkan adalah memiliki pemahaman yang jelas tentang bagaimana informasi yang dikumpulkan dari kuesioner akan dianalisis dan digunakan. Masalah ini harus diatasi sebelum kuesioner dibagikan, karena sudah terlambat sesudahnya.

Gaya pertanyaan harus relatif konsisten, sehingga responden tidak perlu membaca instruksi untuk setiap pertanyaan sebelum menjawabnya. Umumnya merupakan praktik yang baik untuk mengelompokkan pertanyaan terkait bersama-sama agar lebih mudah dijawab. Beberapa ahli menyarankan bahwa kuesioner harus dimulai dengan pertanyaan-pertanyaan yang penting bagi responden, sehingga kuesioner tersebut segera menarik minat mereka dan mendorong mereka untuk menjawabnya. Mungkin langkah yang paling penting adalah meminta beberapa kolega meninjau kuesioner dan kemudian mengujinya terlebih dahulu dengan beberapa orang yang diambil dari kelompok yang akan dikirim kuesioner itu. Sungguh mengejutkan betapa seringnya pertanyaan yang tampaknya sederhana dapat disalahpahami.

Masalah utama dalam mengelola kuesioner adalah membuat peserta mengisi kuesioner dan mengirimkannya kembali. Lusinan buku riset pemasaran telah ditulis tentang cara-cara meningkatkan tingkat respons. Teknik yang umum digunakan termasuk menjelaskan dengan jelas mengapa kuesioner dilakukan dan mengapa responden telah dipilih, menyatakan tanggal kuesioner dikembalikan, menawarkan bujukan untuk mengisi kuesioner (misalnya, pena gratis), dan penawaran untuk memberikan ringkasan tanggapan kuesioner.

Analisis sistem memiliki teknik tambahan untuk meningkatkan tingkat respons di dalam organisasi, seperti membagikan kuesioner secara pribadi dan secara pribadi menghubungi mereka yang belum mengembalikannya setelah satu atau dua minggu, serta meminta supervisor responden untuk mengelola kuesioner di pertemuan kelompok.

Akan sangat membantu untuk memproses kuesioner yang dikembalikan dan mengembangkan laporan kuesioner segera setelah tenggat waktu

kuesioner. Hal ini memastikan bahwa proses analisis berlangsung tepat waktu dan responden yang meminta salinan hasil akan segera menerimanya.

c. Observasi

Observasi atau pengamatan adalah tindakan mengamati proses yang dilakukan, merupakan alat yang ampuh untuk mengumpulkan informasi tentang sistem apa adanya karena memungkinkan analis untuk melihat realitas situasi, daripada mendengarkan orang lain menggambarkannya dalam wawancara. Beberapa studi penelitian menunjukkan bahwa banyak manajer benar-benar tidak ingat bagaimana mereka bekerja dan bagaimana mereka mengalokasikan waktu mereka. Observasi adalah cara yang baik untuk memeriksa validitas informasi yang dikumpulkan dari sumber tidak langsung seperti wawancara dan kuesioner.

Dalam banyak hal, analis menjadi antropolog saat dia berjalan melalui organisasi dan mengamati sistem bisnis sebagaimana fungsinya. Tujuannya adalah untuk menjaga profil tetap rendah, tidak mengganggu mereka yang bekerja, dan tidak memengaruhi mereka yang diamati. Meskipun demikian, penting untuk dipahami bahwa apa yang diamati oleh analis mungkin bukan rutinitas sehari-hari yang normal karena orang cenderung sangat berhati-hati dalam perilakunya saat diawasi. Meskipun praktik normal mungkin melanggar aturan organisasi formal, pengamat tidak mungkin melihat ini. (Ingat bagaimana Anda mengemudi terakhir kali mobil polisi mengikuti Anda?) Jadi, apa yang Anda lihat mungkin bukan yang Anda dapatkan.

Pengamatan sering digunakan untuk melengkapi informasi wawancara. Lokasi kantor seseorang dan perabotannya memberikan petunjuk tentang kekuatan dan pengaruh orang tersebut dalam organisasi dan dapat digunakan untuk mendukung atau menyangkal informasi yang diberikan dalam wawancara. Misalnya, seorang analis mungkin menjadi skeptis terhadap seseorang yang mengklaim menggunakan sistem komputer yang ada secara ekstensif jika komputer tidak pernah dihidupkan saat analis berkunjung. Dalam kebanyakan kasus, observasi mendukung informasi yang diberikan pengguna dalam wawancara. Jika tidak, ini merupakan sinyal penting bahwa kehati-hatian ekstra harus dilakukan dalam menganalisis sistem bisnis.

C. SOAL LATIHAN/TUGAS

1. Jelaskan apa yang dimaksud definisi kebutuhan dan sebutkan tahapan-tahapan dalam fase analisa sistem!
2. Bedakan antara kebutuhan fungsional dan non fungsional!
3. Diskusikan bagaimana menggunakan wawancara untuk mengumpulkan data!
4. Diskusikan bagaimana menggunakan kuesioner untuk mengumpulkan data!.
5. Diskusikan bagaimana menggunakan observasi untuk mengumpulkan data!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Dr. Jawahar. *System Requirement Specifications & Analysis*. Diakses dari:
<http://www.ddegjust.ac.in/studymaterial/pgdca/ms-04.pdf>

Langer, Arthur M. (2008). *Analysis and Design of Information Systems 3rd edition*. Switzerland: Springer.

GLOSARIUM

Data adalah kumpulan angka dan karakter yang tidak berarti. Data tersebut dapat diolah untuk menghasilkan informasi.

Data Flow Diagram (DFD) adalah alat dalam perancangan sistem yang menggunakan simbol untuk menggambarkan aliran data melalui serangkaian proses yang saling berhubungan.

Internet Explorer adalah *browser web* dan perangkat lunak tidak bebas tersedia secara gratis dari Microsoft dan telah disediakan dengan setiap rilis sistem operasi Microsoft Windows sejak 1995.

Microsoft Edge awalnya dikembangkan dengan nama kode Project Spartan, ini adalah *browser web* yang dikembangkan oleh Microsoft dan disertakan dalam sistem operasi Windows.

Microsoft Visual Basic atau VB adalah bahasa pemrograman yang menyediakan lingkungan pengembangan terintegrasi visual (IDE) untuk membuat program perangkat lunak berbasis sistem operasi Microsoft Windows menggunakan model pemrograman (COM).

Microsoft Visual C++ adalah produk Integrated Development Environment atau IDE yang dikembangkan oleh Microsoft untuk bahasa pemrograman C dan C ++.

Microsoft Visual Studio merupakan perangkat lunak (kit) lengkap yang dapat digunakan untuk mengembangkan aplikasi dalam bentuk aplikasi konsol, aplikasi Windows atau aplikasi Web, termasuk aplikasi bisnis, aplikasi pribadi atau komponen aplikasi.

Prototyping adalah sebuah metode siklus hidup sistem berdasarkan konsep model kerja.

Sequence Diagram adalah salah satu diagram di UML, yang menggambarkan kolaborasi dinamis antara banyak objek.

Use case adalah teknik yang digunakan untuk mengembangkan perangkat lunak atau sistem informasi untuk menangkap persyaratan fungsional sistem yang relevan. Kasus penggunaan menggambarkan interaksi yang terjadi antara "partisipan" - pemrakarsa interaksi antara sistem itu sendiri dan sistem yang ada. Situasi tersebut diwakili oleh sebuah serangkaian langkah sederhana.

PERTEMUAN 5

MODELING PADA DFD

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang apa itu pengertian Proses Modelling, Data Flow Diagram, Komponen Data Flow Diagram, dan Bentuk Data Flow Diagram. Dari pertemuan ini diharapkan mahasiswa mampu memahami proses modelling dengan menggunakan DFD.

B. URAIAN MATERI

1. Pengertian Proses Modelling

Model sistem memiliki peran yang penting dalam pengembangan sistem. Sebagai bentuk analisis atau pengguna sistem, akan terus-menerus menangani masalah yang tidak terstruktur. Salah satu cara untuk menyusun masalah seperti itu adalah dengan penggambaran model. Model adalah sebuah bentuk representasi bergambar dari realitas. Model dapat dibangun untuk sistem yang ada sebagai cara untuk lebih memahami sistem tersebut atau untuk sistem yang diusulkan sebagai cara dalam mendokumentasikan persyaratan bisnis atau desain teknis. Konsep penting adalah perbedaan antara model logis dan fisik.

Model logis menunjukkan bagaimana suatu sistem sedang atau apa yang dilakukannya. Model logis adalah sebuah implementasi yang independen, yaitu dengan menggambarkan sistem yang tidak bergantung pada implementasi teknis apa pun. Dengan demikian, model logis mengilustrasikan esensi dari sistem. Sedangkan pada model fisik menunjukkan tidak hanya sistem yang sedang berjalan tetapi juga sistem diimplementasikan secara fisik dan teknis. Model fisik bergantung pada implementasi karena mereka mencerminkan pilihan teknologi dan imitasi dari pilihan teknologi.

Sistem Analisis telah lama menyadari pentingnya memisahkan urusan bisnis dan teknis. Hal ini menjadikan penggunaan model sistem logis untuk menggambarkan kebutuhan bisnis dan model sistem fisik untuk

menggambarkan desain teknis. Kegiatan analisis sistem cenderung berfokus pada model sistem logis karena alasan berikut:

- a. Model logis menghilangkan bias yang merupakan hasil dari cara sistem saat ini yang diimplementasikan dengan cara orang berpikir sistem mungkin diimplementasikan. Akibatnya, model logis mendorong kreativitas.
- b. Model logis mengurangi risiko kehilangan persyaratan bisnis karena disibukkan dengan detail teknis. Kesalahan seperti itu bisa sulit untuk diperbaiki setelah sistem diimplementasikan. Dengan memisahkan apa yang harus dilakukan sistem dari bagaimana sistem akan melakukannya, hal tersebut dapat menganalisis persyaratan kelengkapan, akurasi, dan konsistensi dengan lebih baik.
- c. Model logis memungkinkan komunikasi dengan pengguna akhir dalam bahasa non-teknis atau kurang teknis. Dengan demikian, tidak akan kehilangan persyaratan dalam jargon teknis dari disiplin komputasi.

Pembahasa ini akan berfokus secara eksklusif pada proses modeling / proses pemodelan logis pada sistem analis. Proses Modeling adalah suatu teknologi yang digunakan untuk mengelola dan merekam proses dalam suatu sistem. Teknologi yang digunakan adalah untuk mengelola dan merekam struktur dan proses data melalui sistem logika, strategi, dan proses yang akan direalisasikan oleh proses sistem.

Dalam konteks sistem informasi, proses model logis digunakan untuk mendokumentasikan fokus proses sistem informasi dari perspektif pengguna sistem. Perhatikan pada jenis model yang digunakan dalam proses modeling adalah menggunakan diagram alir data yaitu Data flow diagram. Data Flow Diagram menggambarkan fokus komunikasi dari perspektif pemilik dan pengguna sistem. Proses modeling ini berasal dari metode rekayasa perangkat lunak. Banyak yang telah menemukan berbagai jenis proses modeling, seperti bagan struktur program, flowchart logika, atau tabel keputusan dalam bidang aplikasi pemrograman.

Ada banyak teknik proses modeling yang digunakan saat ini. Dalam bab ini, pembahasan akan berfokus pada salah satu teknik yang paling umum digunakan yaitu data flow diagram. Terdapat juga bentuk dari data flow diagram yaitu DFD logis dan DFD fisik.

2. Data Flow Diagram

Data Flow Diagram adalah analisis sistem yang dapat menyusun representasi grafis dari proses data di seluruh organisasi, atau Data Flow Diagram dapat disebut dengan alat yang menggambarkan aliran data melalui suatu sistem data dan pekerjaan pada pemrosesan yang dilakukan oleh sistem tersebut. persamaannya meliputi diagram gelembung, grafik transformasi, dan proses model. Alat perencanaan DFD disebut dengan diagram dekomposisi.

Definisi kunci dari data flow diagram adalah diagram yang terdiri dari proses dan menggambarkan ruang lingkup sistem. Data flow diagram merupakan level tertinggi yang mendeskripsikan semua input atau output sistem dalam sistem. Data flow chart hanya memiliki proses dan tidak boleh memiliki komponen penyimpanan (storage), disebut juga data level 0.

3. Kegunaan DFD

Kegunaan dari Data Flow Diagram (DFD) adalah untuk menyediakan hubungan semantik antara pengguna dan pengembang sistem. Gambar ini:

- a. Secara grafik, menghilangkan ribuan kata
- b. Mewakili hubungan logis dan membuat model "apa yang dilakukan sistem", bukan hanya model yang secara fisik menampilkan "bagaimana model dieksekusi".
- c. Penciptaan hierarki untuk menunjukkan sistem secara detail.
- d. Beberapa simbol yang digunakan memudahkan pengguna untuk memahami dan melihatnya.

Tujuan DFD adalah untuk memahami model sistem secara kasar. Diagram ini adalah dasar dari analisis struktur sistem. Analisis struktur sistem lainnya (seperti diagram struktur data, kamus data) dan teknologi lain (seperti tabel keputusan atau pohon keputusan) mendukung DFD.

4. Syarat-syarat Pembuatan Data Flow Diagram

Persyaratan untuk membuat DFD ini akan membantu pemrogram menghindari pembuatan DFD yang salah atau DFD yang tidak sesuai secara logis. Beberapa syarat pembuatan DFD dapat membantu programmer membentuk DFD yang benar, menampilkan gambar yang ingin dipahami dan gambar yang mudah dibaca pengguna. Persyaratan pembuatan DFD ini

adalah:

- a. Berikan nama yang jelas untuk setiap komponen DFD.
- b. Beri nomor pada komponen proses.
- c. Jelaskan DFD se jelas mungkin agar dapat dipahami.
- d. Hindari skema gambar DFD yang rumit.
- e. Pastikan untuk mendeskripsikan DFD dengan cara yang logis.

5. Penggambaran DFD

Pada saat menggambar DFD memang tidak ada aturan pasti. Berdasarkan berbagai referensi yang ada, langkah-langkah pembuatan DFD biasanya adalah sebagai berikut:

- a. Pertama tentukan semua entitas eksternal yang terlibat dalam sistem.
- b. Identifikasi semua input dan output yang terkait dengan entitas eksternal pada sistem.
- c. Gambarlah diagram konteks (context diagram) Diagram ini adalah diagram level tertinggi dari DFD, yang menggambarkan hubungan antara sistem dan lingkungan luar. Caranya seperti dibawah ini:
 - 1) Menetapkan nama dari sistemnya.
 - 2) Menetapkan batasan dari sistemnya.
 - 3) Menetapkan terminator apa saja yang ada dalam sistem.
 - 4) Menetapkan apa yang diterima atau diberikan terminator ke system tersebut.
 - 5) Buat gambar diagram konteks.
- d. Buat Diagram Level Zero

Diagram ini merupakan analisis dari diagram konteks. Caranya sebagai berikut:

- 1) Menentukan proses utama yang ada pada system tersebut.
- 2) Menentukan kebutuhan sistem atau kebutuhan setiap proses, dan memperhatikan konsep keseimbangan (arus data keluar atau masuk satu tingkat, arus data harus sama dengan arus data masuk atau keluar tingkat berikutnya).
- 3) Jika dibutuhkan, Menampilkan penyimpanan data (database primer) sebagai sumber dan target arus data.
- 4) Gambarkan diagram level zero.
 - a) Hindari perpotongan aliran data.

- b) Berikan nomor ke proses utama (nomor tidak menunjukkan urutan proses)..

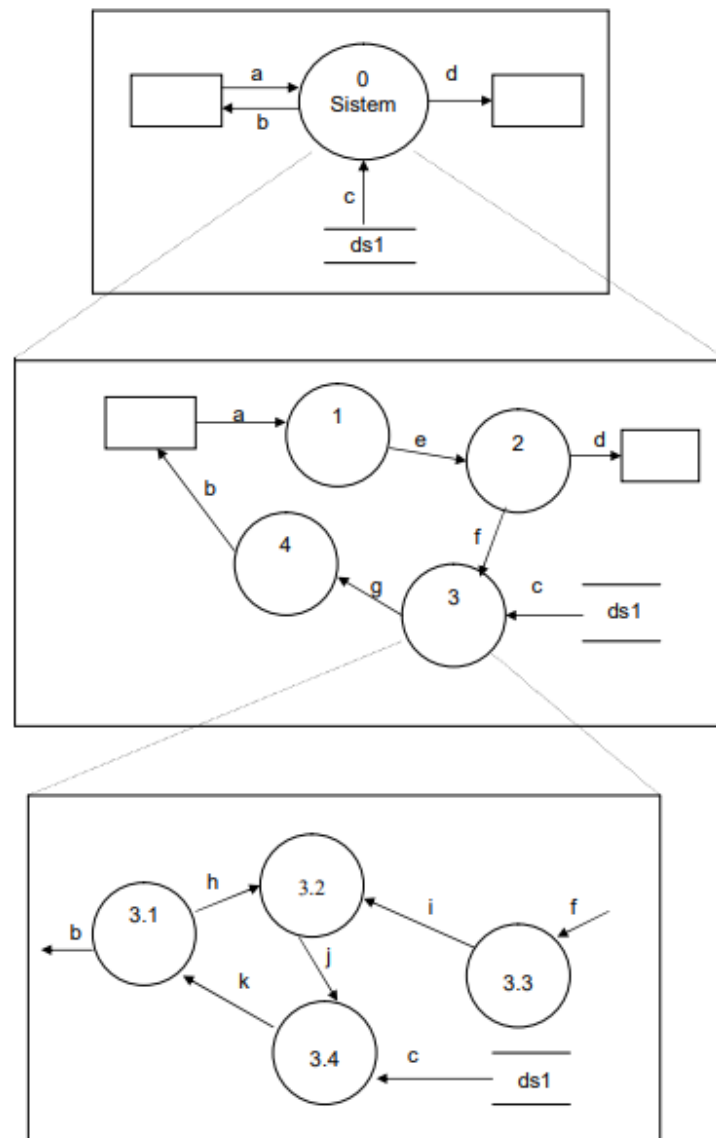
e. Buat Diagram Level Satu

Diagram ini merupakan analisis dari diagram level zero. Caranya sebagai berikut :

- 1) Identifikasi proses (subproses) yang lebih kecil dari proses utama tingkat nol.
- 2) Tentukan apa yang diberikan atau diterima setiap sub-proses untuk memahami dari sistem dan memperhatikan konsep keseimbangan.
- 3) Jika perlu, tampilkan penyimpanan data (transaksi) sebagai sumber dan tujuan aliran data.
- 4) Jelaskan DFD level 1
 - a) Hindari perpotongan aliran data.
 - b) Beri nomor pada setiap sub-proses untuk menunjukkan rincian dari proses sebelumnya.

f. DFD Level Dua, Tiga, ...









Diagram ini merupakan terobosan dari level sebelumnya. Lakukan proses analisis hingga siap menggunakannya dalam program. Aturan yang digunakan sama dengan level pertama.



Gambar 1. Levelisasi DFD

6. Komponen Data Flow Diagram

Ada empat simbol dalam bahasa DFD (proses, aliran data, penyimpanan data, dan entitas eksternal), yang masing-masing diwakili oleh simbol grafik yang berbeda. Berikut adalah gambar dari simbol komponen data flow diagram :

Simbol	Arti	Contoh
	Entitas (<i>Entity</i>)	
	Aliran Data (<i>Data Flow</i>)	
	Proses	
	Penyimpanan Data (<i>Data Store</i>)	

Gambar 2. Komponen DFD

Terdapat 4 komponen data flow diagram terdiri dari :

a. Entitas eksternal

Entitas eksternal adalah individu, organisasi, unit organisasi, atau sistem di luar sistem, tetapi berinteraksi dengan mereka (misalnya, pelanggan, lembaga kliring, organisasi pemerintah, sistem akuntansi). Entitas eksternal biasanya sesuai dengan peran utama yang diidentifikasi dalam use case. Entitas eksternal menyediakan data atau menerima data dari sistem, dan fungsi eksternalnya adalah untuk menetapkan batasan sistem. Setiap entitas eksternal memiliki nama dan deskripsi.

Intinya dari entitas eksternal yang perlu diingat adalah bahwa mereka berada di luar sistem, tetapi mungkin atau mungkin bukan bagian dari organisasi. Orang yang menggunakan informasi dari sistem untuk melakukan proses lain atau memutuskan informasi mana yang masuk ke sistem dicatat sebagai entitas eksternal (misalnya, manajer, staff).

b. Data Flow

Data Flow adalah bagian dari data (misalnya, kuantitas yang tersedia) (kadang-kadang disebut elemen data), atau kumpulan logis dari beberapa informasi (misalnya, permintaan bahan kimia baru). Setiap aliran data harus dinamai menurut kata benda. Deskripsi aliran data mencantumkan dengan tepat elemen data mana yang dikandung aliran. Aliran data diwakili oleh panah, yang menunjukkan arah proses masuk dan keluar.

Data Flow ini digunakan untuk menjelaskan perpindahan data atau paket data / informasi dari satu bagian sistem ke bagian lain. Selain menampilkan arah, aliran data dalam model yang dibuat oleh profesional sistem juga dapat merepresentasikan pesan, tabel, bilangan real, dan berbagai informasi terkait komputer. Data Flow juga dapat

merepresentasikan data / informasi yang tidak ada hubungannya dengan komputer. Data Flow perlu diberi nama sesuai dengan data / informasi yang terlibat, dan kata benda (seperti laporan penjualan) biasanya digunakan untuk memberi nama aliran data. Aliran data akan selalu datang dari atau masuk ke proses, dan tanda panah dengan panah menunjukkan arah masuk atau keluar dari proses.

Aliran data menunjukkan apa yang dimasukkan dalam setiap proses dan keluaran apa yang dihasilkan oleh setiap proses. Setiap proses harus membuat setidaknya satu aliran data keluaran, karena jika tidak ada keluaran, proses tidak melakukan apa pun. Demikian pula, setiap proses memiliki setidaknya satu aliran data masukan, karena sulit menghasilkan keluaran tanpa masukan, jika bukan tanpa kesulitan.

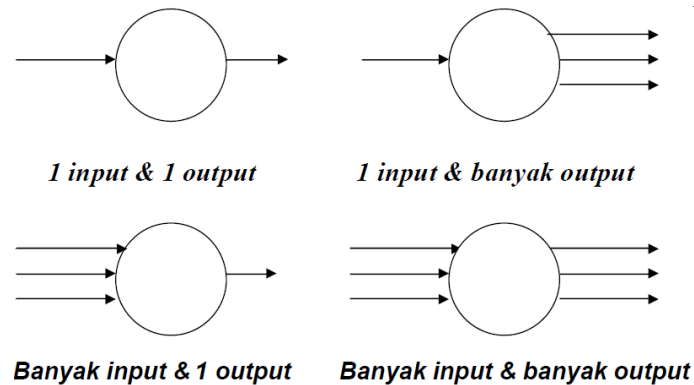
c. Proses

Proses adalah proses Proses adalah aktivitas atau fungsi yang dilakukan untuk beberapa alasan bisnis. Prosesnya bisa manual atau komputerisasi. Setiap proses harus dimulai dengan kata kerja dan diakhiri dengan kata benda (misalnya, "menentukan permintaan"), dan nama harus pendek tetapi berisi informasi yang cukup sehingga pembaca dapat dengan mudah memahami apa yang mereka lakukan.

Umumnya, setiap proses hanya dapat melakukan satu aktivitas, sehingga sebagian besar analisis sistem menghindari penggunaan kata "dan" dalam nama proses karena itu berarti proses tersebut melakukan banyak aktivitas. Selain itu, setiap proses harus memiliki setidaknya satu aliran data masukan dan setidaknya satu aliran data keluaran. Komponen proses menggambarkan bagian dari sistem yang mengubah masukan menjadi keluaran.

Sebutkan proses untuk mendeskripsikan proses / aktivitas yang sedang / akan dilaksanakan. Proses penamaan dilakukan dengan menggunakan verba transitif (verba yang membutuhkan benda), seperti menghitung gaji, mencetak KRS, dan menghitung jumlah SKS. Yang harus diperhatikan dalam proses adalah proses tersebut harus memiliki input dan output. Proses dapat dihubungkan ke komponen terminator, penyimpanan data atau proses melalui aliran data, dan tidak boleh ada proses dengan nama yang sama.

Berikut kemungkinan yang dapat terjadi dalam proses sehubungan dengan input dan output :



Gambar 3. Proses Input & Output

d. Data Store

Data Store atau penyimpanan data adalah kumpulan data yang disimpan dengan cara tertentu (yang mana ditentukan kemudian saat membuat model fisik). Setiap penyimpanan data diberi nama dengan kata benda dan diberi nomor identifikasi dan deskripsi. Penyimpanan data membentuk titik awal untuk model data dan merupakan penghubung utama antara model proses dan model data.

Arus data yang keluar dari suatu penyimpanan data menunjukkan bahwa informasi telah diambil dari penyimpanan data. Komponen ini digunakan untuk memodelkan satu set paket data dan dinamai beberapa kata benda, seperti Student. Penyimpanan data biasanya berhubungan dengan penyimpanan, seperti file atau database yang berhubungan dengan penyimpanan komputer, seperti file floppy disk, file hard disk, dan file tape.

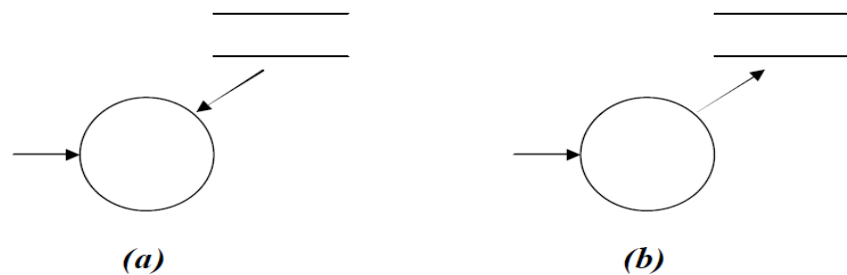
Penyimpanan data juga menangani penyimpanan manual, seperti buku alamat, folder, dan agenda. Penyimpanan data hanya berhubungan dengan arus data pada komponen proses, tidak dengan komponen DFD lainnya.

Aliran data yang menghubungkan penyimpanan data ke proses memiliki arti sebagai berikut:

- 1) Arus data dari penyimpanan data, artinya membaca atau mengakses satu paket data, lebih dari satu paket data, bagian dari satu paket data,

atau bagian dari lebih dari satu paket data untuk suatu proses (lihat Gambar (a)).

- 2) Arus data ke area penyimpanan data, yang berarti memperbarui data, seperti menambah satu atau lebih paket data baru, menghapus satu atau lebih paket atau mengubah / memodifikasi satu atau lebih paket data (lihat Gambar (b)).



Gambar 4. Alur Data Store

7. Bentuk Data Flow Diagram

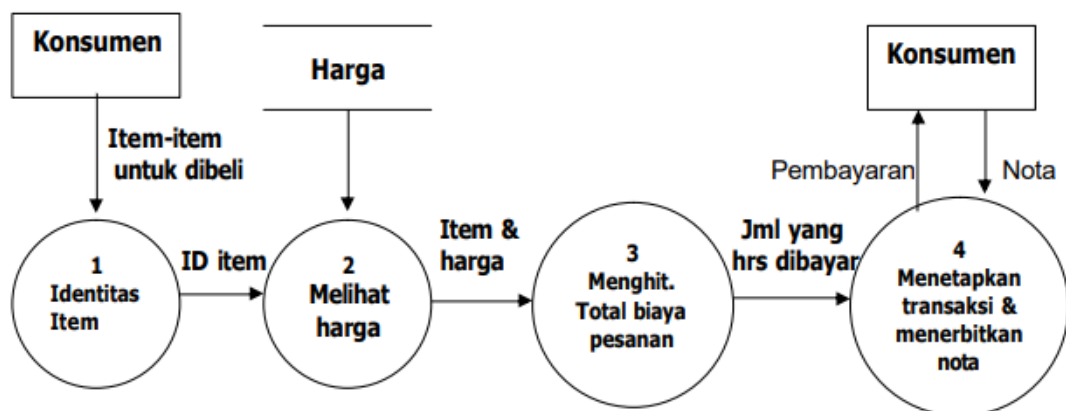
a. Jenis Data Flow Diagram

Ada 2 bentuk dalam Data Flow Diagram (Diagram Alir), yaitu DFD Fisik (diagram aliran data fisik) dan DFD Logik (diagram aliran data logis). DFD fisik mengacu pada implementasi proses sistem, sedangkan DFD Logika lebih memperhatikan proses yang terdapat dalam sistem. Ini seperti proses yang dibutuhkan sistem secara logis. Karena sistem yang diusulkan belum tentu dapat diterima oleh pengguna sistem, dan biasanya sistem yang diusulkan terdiri dari beberapa opsi, dibandingkan dengan PDFD, lebih penting untuk melakukan penggambaran logis sebelumnya tanpa mementingkan aplikasi fisiknya. Efektif, sekaligus menghemat waktu menggambar. Untuk sistem komputer, deskripsi LDFD secara logis hanya berfokus pada persyaratan proses dari sistem yang diusulkan, dan proses yang dijelaskan biasanya hanya proses komputer.

b. Physical Data Flow Diagram

DFD Fisik, menjelaskan di mana dan siapa yang akan melakukan proses, dan daftar teknik khusus yang digunakan untuk melakukan proses tersebut. Untuk memahami bagaimana sistem diimplementasikan, PDFD harus memiliki persyaratan berikut:

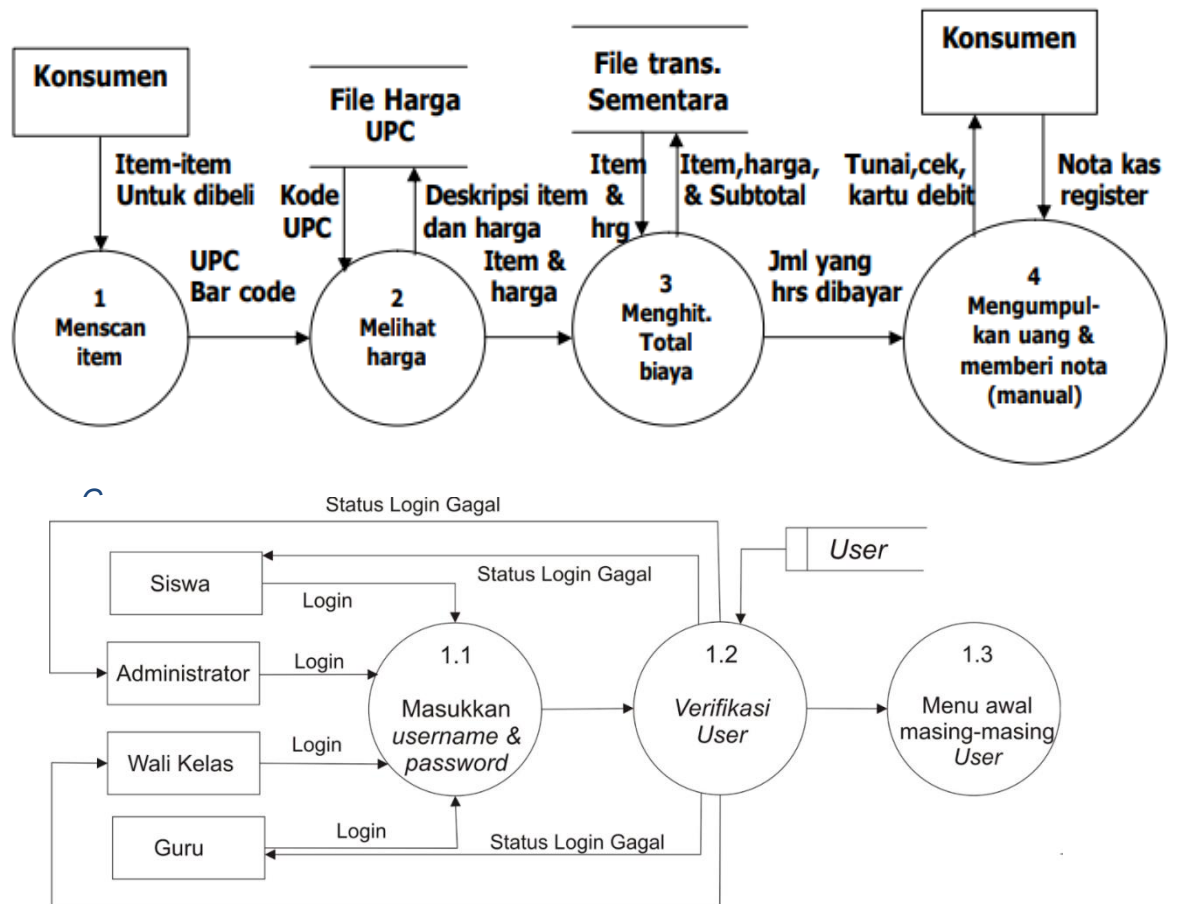
- 1) Juga menjelaskan proses yang masih dilakukan secara manual.
- 2) Nama aliran data harus menggambarkan fakta penerapannya, seperti nomor tabel dan media. Nama aliran data juga menjelaskan waktu aliran (misalnya, jam atau hari). Oleh karena itu, nama aliran data harus memiliki informasi yang cukup rinci untuk menunjukkan bagaimana pengguna sistem memahami pekerjaan sistem.
- 3) Area penyimpanan data dapat menggambarkan area penyimpanan non-komputer, seperti kotak input / output, yang digunakan sebagai buffer untuk proses bersamaan yang berjalan pada kecepatan berbeda, sehingga akan ada data yang menunggu di buffer.
- 4) Nama penyimpanan data harus menggambarkan jenis aplikasi yang manual atau terkomputerisasi. Manual disini seperti menunjukkan notebook, workbench atau in / out box. Terkomputerisasi, seperti menampilkan file sequence, file ISAM, file database, dll.
- 5) Proses harus mendeskripsikan nama prosesor, yaitu nama orang, departemen, sistem komputer, atau program komputer yang melakukan proses.



Gambar 5. Contoh Physical Data Flow Diagram

c. Logical Data Flow Diagram

Logika DFD menampilkan aliran data tanpa melihat kapan aliran data tersebut terjadi. Diagram aliran data logis hanya berfokus pada proses-proses yang berlangsung di sistem, tanpa mempertimbangkan media fisik yang digunakan untuk memindahkan data.



Contoh Logical Data Flow Diagram

C. SOAL LATIHAN/TUGAS

1. Menurutmu apa itu proses modelling itu!
2. Gambarkan DFD suatu sistem!
3. Berikan contoh lain dari Physical Data Flow Diagram dan Logical Data Flow Diagram!
4. Ada berapa komponen pada Data Flow Diagram? Jelaskan!
5. Jelaskan secara singkat alur dari DFD ini!

D. REFERENSI

- Burch, J.G., System, Analysis, Design, and Implementation, Boyd & Fraser Publishing Company, 1992.
- John G. Burch, Jr, Felix R. Strater, Gary Grudnitski, Information Systems : Theory and Practice, Second Edition, John Wiley & Sons, 1979.
- Meilir Page-Jones, The Practical Guide to Structured Systems Design, Second Edition, Yourdon Press, Prentice Hall, 1988.
- I.T. Hawryszkiewicz, Introduction Systems Analysis and Design, Second Edition, Prentice Hall, 1991
- Raymond McLeod, Jr, Management Information System : A Study of Computer-Based Information Systems, Sixth Edition, Prentice Hall, 1979
- A. Ziya Aktas, Structured Analysis & Design of Information Systems, NJ: Prentice Hall, 1987, hal. 65
- Dennis, Alan, Wixom, Barbara Haley, Roth, Roberta M. (2013). System Analysis and Design 5th edition. New Jersey: John Willey & Sons, Inc.

GLOSARIUM

Esensi adalah adanya kenyataan, yaitu hakikatnya. Pengertian mengenai esensi mengalami perubahan sesuai dengan konsep penggunaannya, sehingga esensi ialah pada konsepnya sendiri.

Review adalah sebuah ringkasan, ulasan dari beberapa sumber seperti buku, jurnal, film, berita, suatu produk dan lain-lain.

Developer, seseorang yang bertugas membangun sebuah sistem, merancang arsitektur, mengimplimentasikan serta mengembangkan sistem tersebut dimasa yang akan mendatang.

Programmer adalah seseorang yang memiliki kemampuan atau skill menulis dan merancang kode program-program (syntax) komputer

Entitas adalah sesuatu yang memiliki keberadaan yang unik dan berbeda, walaupun tidak harus dalam bentuk fisik.

Terminator adalah simbol untuk menunjukkan awal atau akhir dari aliran proses.

PERTEMUAN 6

FLOWCHART

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian sistem Flowchart, symbol-simbol Flowchart, dan jenis-jenis Flowchart. Dari pertemuan ini diharapkan mahasiswa mampu memahami sistem Flowchart.

B. URAIAN MATERI

1. Pengertian Flowchart

Flowchart merupakan alat yang digunakan untuk serangkaian tindakan ditampilkan dalam proses yang mudah dipahami. Tujuan dari diagram alur ini meliputi:

- a. Tentang cara menyelesaikan proses.
- b. Mempelajari *updating* proses.
- c. Bagaimana melakukan proses komunikasi dengan orang lain.
- d. Perlu komunikasi yang lebih baik antara orang-orang yang terlibat dalam proses yang sama.
- e. Proses perekaman.
- f. Rencanakan aktivitas

2. Pedoman-Pedoman untuk Membuat Flowchart

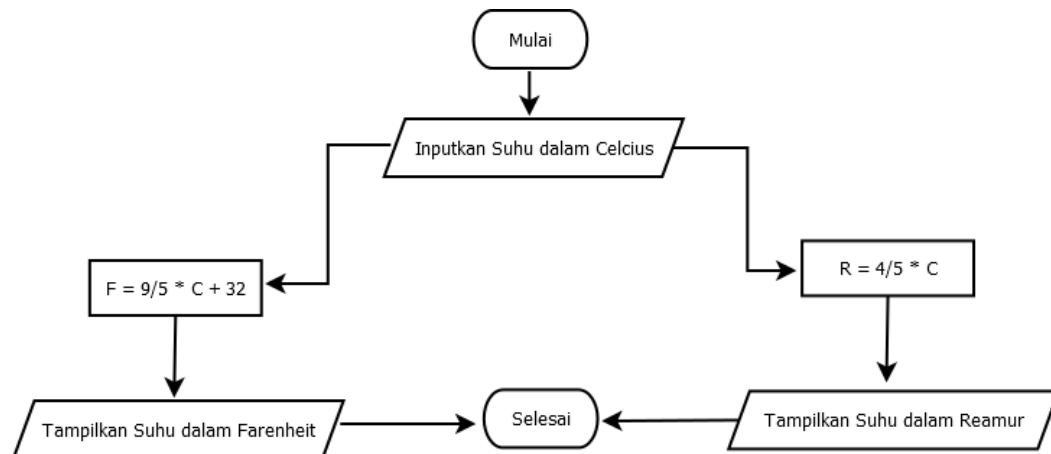
Flowchart adalah diagram yang secara logis menampilkan aliran dari suatu program atau proses sistem. Diagram alir terutama digunakan untuk bantuan komunikasi dan dokumentasi. Saat menggambar diagram alir, analis atau pemrogram sistem dapat mengikuti pedoman berikut.

- a. Flowchart digambar dari atas ke bawah halaman, dari kiri ke kanan.
- b. Kegiatan yang dijelaskan harus didefinisikan dengan cermat dan pembaca harus memahami definisi ini.
- c. Waktu mulai dan berakhir acara harus ditentukan dengan jelas.
- d. Setiap langkah aktivitas harus dijelaskan dengan deskripsi kata kerja,

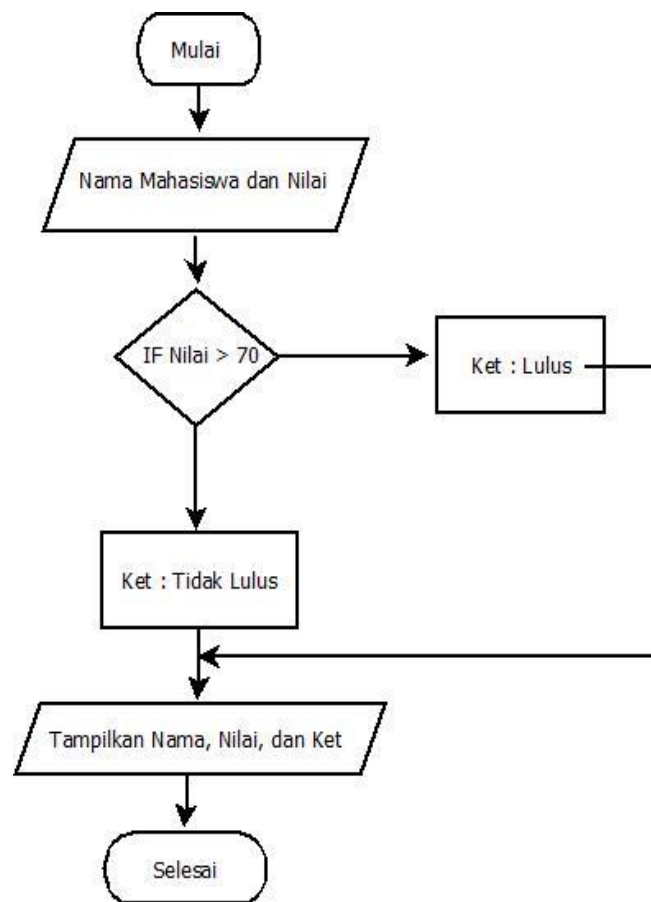
misalnya **“MENGONVERSIKAN SUHU”**, **“MENENTUKAN KELULUSAN MAHASISWA”**.

- e. Setiap langkah aktivitas harus dilakukan dalam urutan yang benar.
- f. Ruang lingkup dan ruang lingkup kegiatan yang dijelaskan harus dilacak dengan cermat. Cabang-cabang yang merentang aktivitas tidak perlu digambar pada diagram alur yang sama. Jika cabang tidak terkait dengan sistem, Anda harus menggunakan simbol konektor dan meletakkan cabang di halaman terpisah, atau menghapus cabang sepenuhnya.
- g. Gunakan simbol diagram alur standar

Saat membuat Flowchart baiknya dibuat bersama team atau kelompok anggota. Karena semua anggota kelompok perlu berdiskusikan dan menyetujui batasan-batasan kegiatan yang akan dimasukkan ke dalam flowchart. Proses tersebut lalu dimasukan ke dalam langkah-langkah yang dapat dituliskan dalam bentuk kata kerja yang singkat dan dapat mudah dimengerti. Langkah-langkah tersebut di tulis di dalam sebuah kotak, kemudian gambarkan dengan menghubungkan tiap kotak dengan tanda panah sesuai urutan langkah-langkah proses (Contoh **Gambar 1 & 2**).



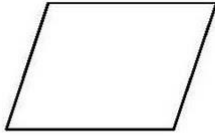
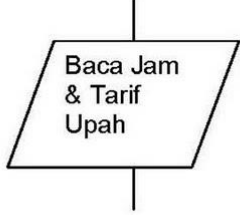

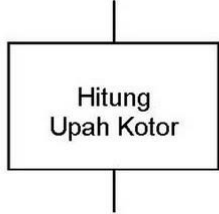

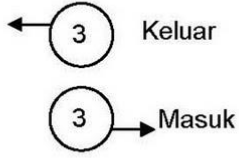

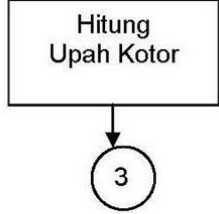
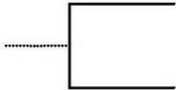
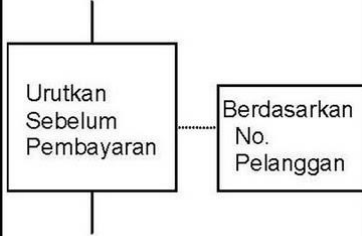
gambar 7. Flowchart Mengonversikan Suhu



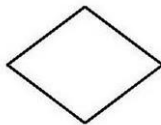
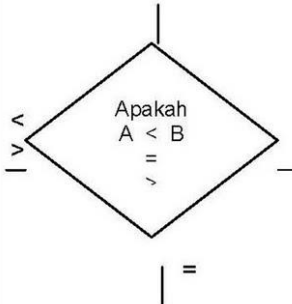



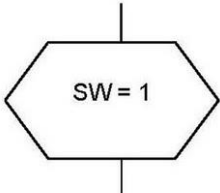

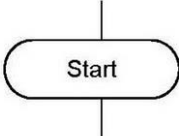

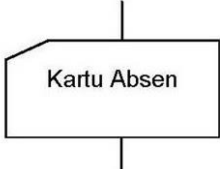
Gambar 8. Flowchart Menentukan Kelulusan Mahasiswa

3. Simbol-Simbol Flowchart



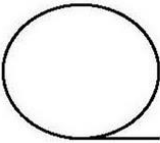
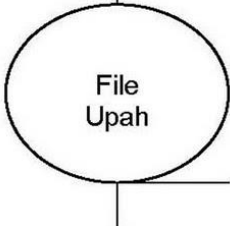
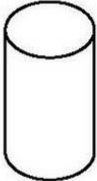

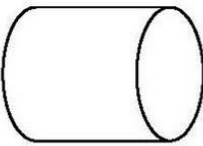
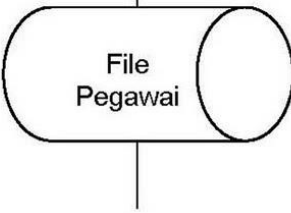
Secara umum, simbol diagram alur yang digunakan oleh Gilbreth tidak terlalu dikenal. Ini mungkin karena penggunaan Microsoft Office yang meluas, di mana Microsoft Office merujuk simbol diagram alur dasar ke simbol diagram alur untuk pemrosesan data. Simbol diagram alur yang umum digunakan adalah simbol diagram alur standar yang dikeluarkan oleh ANSI dan ISO. Simbol tersebut dapat dilihat pada Gambar 2. Simbol diagram alur standar berikut:

SIMBOL	ARTI	CONTOH
Input / Output 	Merepresentasikan Input data atau Output data yang diproses atau Informasi.	
Proses 	Mempresentasikan operasi	
Penghubung 	Keluar ke atau masuk dari bagian lain flowchart khususnya halaman yang sama	
Anak Panah 	Merepresentasikan alur kerja	
Penjelasan 	Digunakan untuk komentar tambahan	


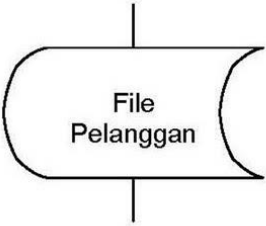
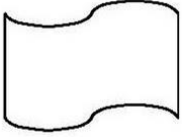
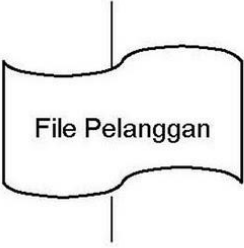
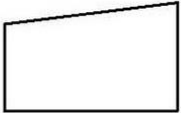




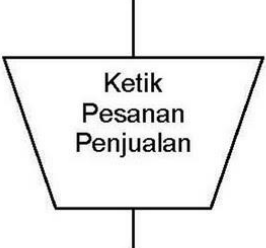
Gambar 9. Simbol-Simbol Flowchart ISO & ANSI

SIMBOL	ARTI	CONTOH
Keputusan 	Keputusan dalam program	
Predefined Process 	Rincian operasi berada di tempat lain	
Preparation 	Pemberian harga awal	
Terminal Points 	Awal / akhir flowchart	
Punched card 	Input / output yang menggunakan kartu berlubang	



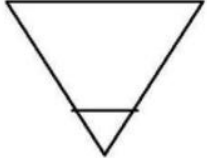
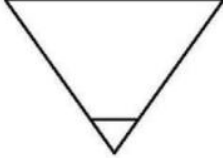
Gambar 10. Simbol-Simbol Flowchart ISO & ANSI (Terusan)

SIMBOL	ARTI	CONTOH
Dokumen 	I/O dalam format yang dicetak	
Magnetic Tape 	I/O yang menggunakan pita magnetik	
Magnetic Disk 	I/O yang menggunakan disk magnetik	
Magnetic Drum 	I/O yang menggunakan drum magnetik	

Gambar 11. Simbol-Simbol Flowchart ISO & ANSI (Terusan)

SIMBOL	ARTI	CONTOH
On-line Storage 	I/O yang menggunakan penyimpanan akses langsung	
Punched Tape 	I/O yang menggunakan pita kertas berlubang	
Manual Input 	Input yang dimasukkan secara manual dari keyboard	
Display 	Output yang ditampilkan pada terminal	
Manual Operation 	Operasi Manual	

Gambar 12. Simbol-Simbol Flowchart ISO & ANSI (Terusan)

SIMBOL	ARTI	CONTOH
Communication Link 	Transmisi data melalui channel komunikasi, seperti telepon	Terminal Komputer 
Off-line Storage 	Penyimpanan yang tidak dapat diakses oleh komputer secara langsung	

Gambar 13. Simbol-Simbol Flowchart ISO & ANSI (Terusan)

4. Jenis-Jenis Flowchart

Pada flowchart terdapat beberapa jenis kategori sesuai dengan fungsi dan proses serta tingkat ketertarikan pengguna. Flowchart terbagi menjadi lima jenis, yaitu:

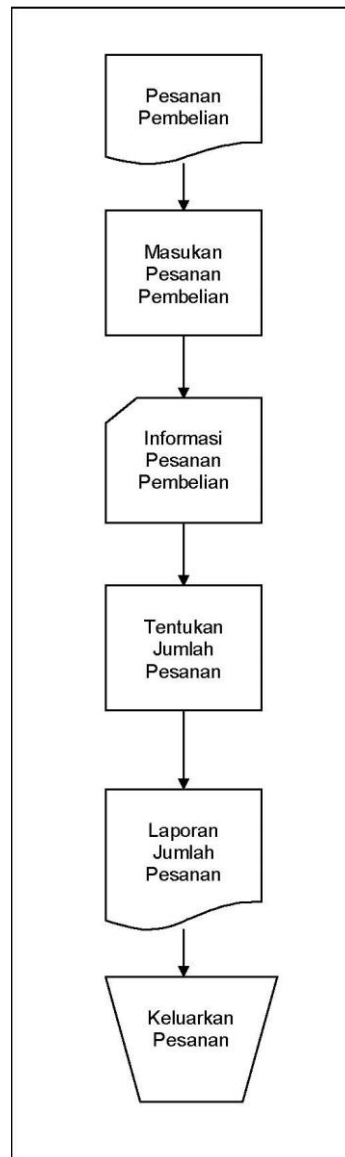
- Flowchart Sistem (System Flowchart)
- Flowchart Paperwork / Flowchart Dokumen (Document Flowchart)
- Flowchart Skematik (Schematic Flowchart)
- Flowchart Program (Program Flowchart)
- Flowchart Proses (Process Flowchart)

Berikut adalah penjelasan terperinci jenis-jenis flowchart tersebut:

a. Flowchart Sistem (System Flowchart)

Flowchart Sistem merupakan diagram yang menunjukkan alur kerja dari keseluruhan sistem atau pekerjaan yang sedang dilakukan, serta menggambarkan urutan proses yang ada pada sistem. Dapat dikatakan bahwa flowchart merupakan gambaran grafis dari urutan proses yang digabungkan membentuk suatu sistem. Diagram alir sistem mencakup data yang mengalir melalui sistem dan proses perubahan data. Contoh diagram alir sistem sederhana dapat dilihat pada Gambar 3, yang melibatkan

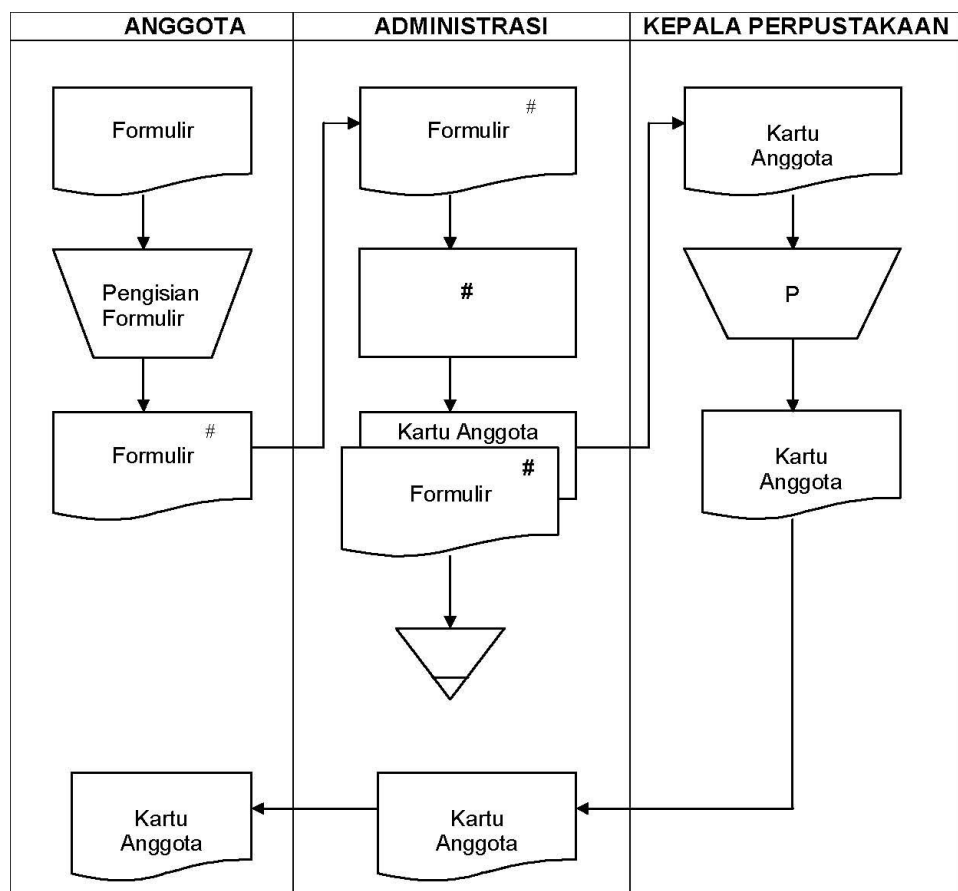
pemesanan kopi di kafe berikut:



Gambar 14. Flowchart Sistem

b. Flowchart Paperwork / Dokumen (Document Flowchart)

Flowchart Dokumen (Paperwork) Merupakan diagram alir yang melacak aliran data yang ditulis melalui sistem. Tujuan utamanya adalah untuk melacak aliran formulir dan laporan sistem dari satu bagian ke bagian lain, serta cara memproses, mencatat, dan menyimpan aliran formulir dan laporan. **Gambar 9.** Mengilustrasikan contoh diagram alur proses pembuatan kartu anggota perpustakaan.



Gambar 15. Flowchart Document/Paperwork

KETERANGAN:

: Memasukan data calon anggota ke dalam computer (proses input data).

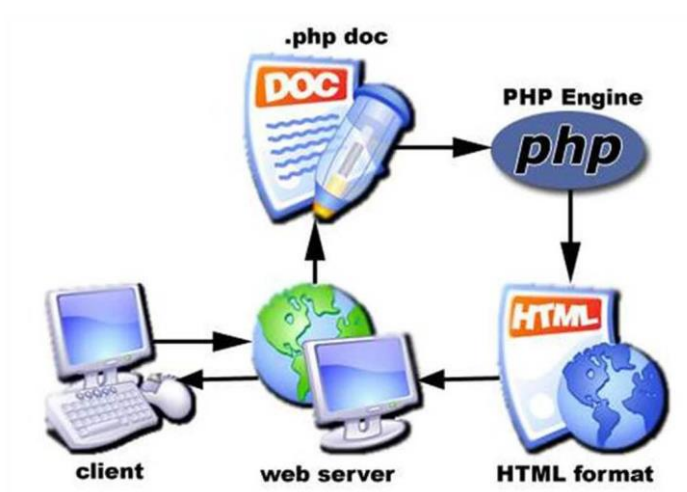
P : Tanda tangan dan validasi data.

c. Flowchart Skematik (Schematic Flowchart)

Flowchart Skematik hampir sama dengan Flowchart Sistem yang menjelaskan sistem atau program. Diagram alir Diagram ini tidak hanya menggunakan simbol diagram alir standar, tetapi juga menggunakan gambar komputer, periferal, tabel, atau peralatan lain yang digunakan dalam sistem.

Diagram alur skematik digunakan sebagai alat komunikasi antara analis sistem dan mereka yang tidak terbiasa dengan simbol diagram alur konvensional. Menggunakan gambar sebagai pengganti simbol diagram alur akan menghemat waktu orang untuk memahami diagram alur sebelum mereka mempelajari simbol abstrak.

Gambar-gambar ini akan mengurangi kemungkinan kesalahpahaman sistem di kemudian hari akibat tidak memahami simbol-simbol yang digunakan. Gambar-gambar ini juga memudahkan pengamat untuk memahami segala sesuatu yang diinginkan analis, sehingga hasilnya lebih efektif dan tidak ada kesalahpahaman. **Gambar 16** menunjukkan contoh flowchart skematik.



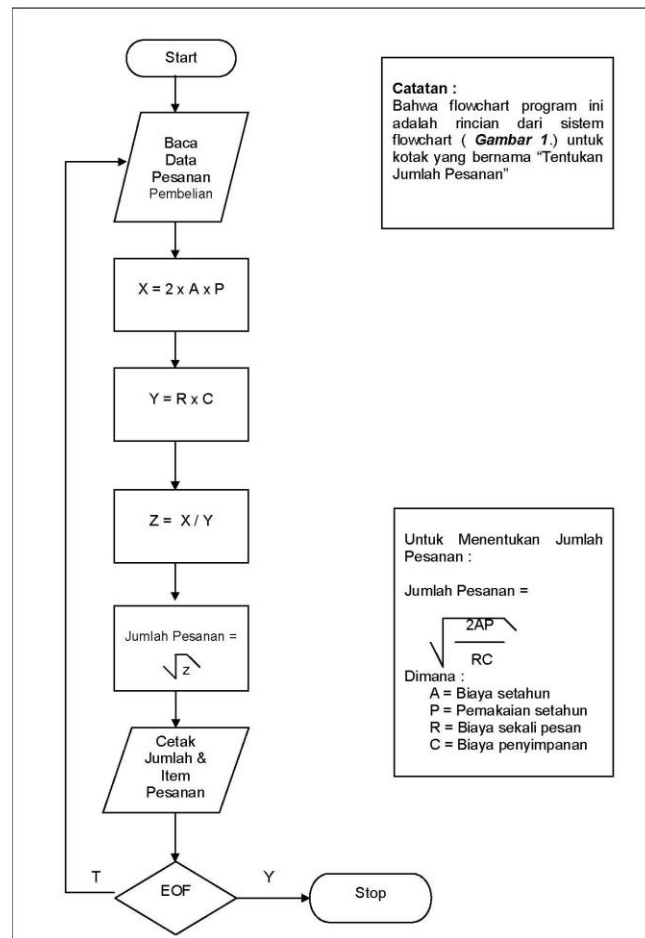
Gambar 16. Flowchart Skematik

d. Flowchart Program (Program Flowchart)

Program diagram alir adalah hasil dari diagram alir sistem. Diagram alir program adalah penjelasan rinci tentang bagaimana sebenarnya menjalankan setiap langkah program atau proses. Flowchart

menggambarkan urutan setiap program atau proses dalam urutan yang benar.

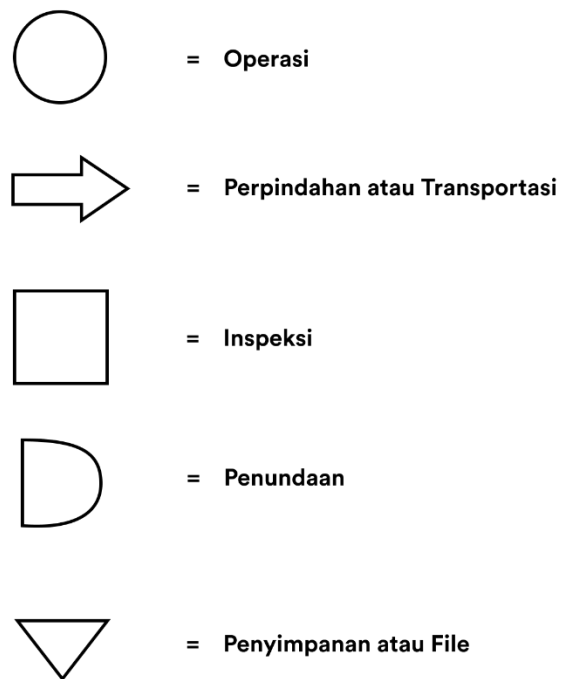
Pemrogram menggunakan diagram alir program untuk menggambarkan urutan instruksi dari program komputer. Analis Sistem menggunakan diagram alir program untuk menggambarkan urutan tugas pekerjaan dalam suatu proses atau operasi. Contoh diagram alir program dapat dilihat pada **Gambar 17**:



Gambar 17. Flowchart Program

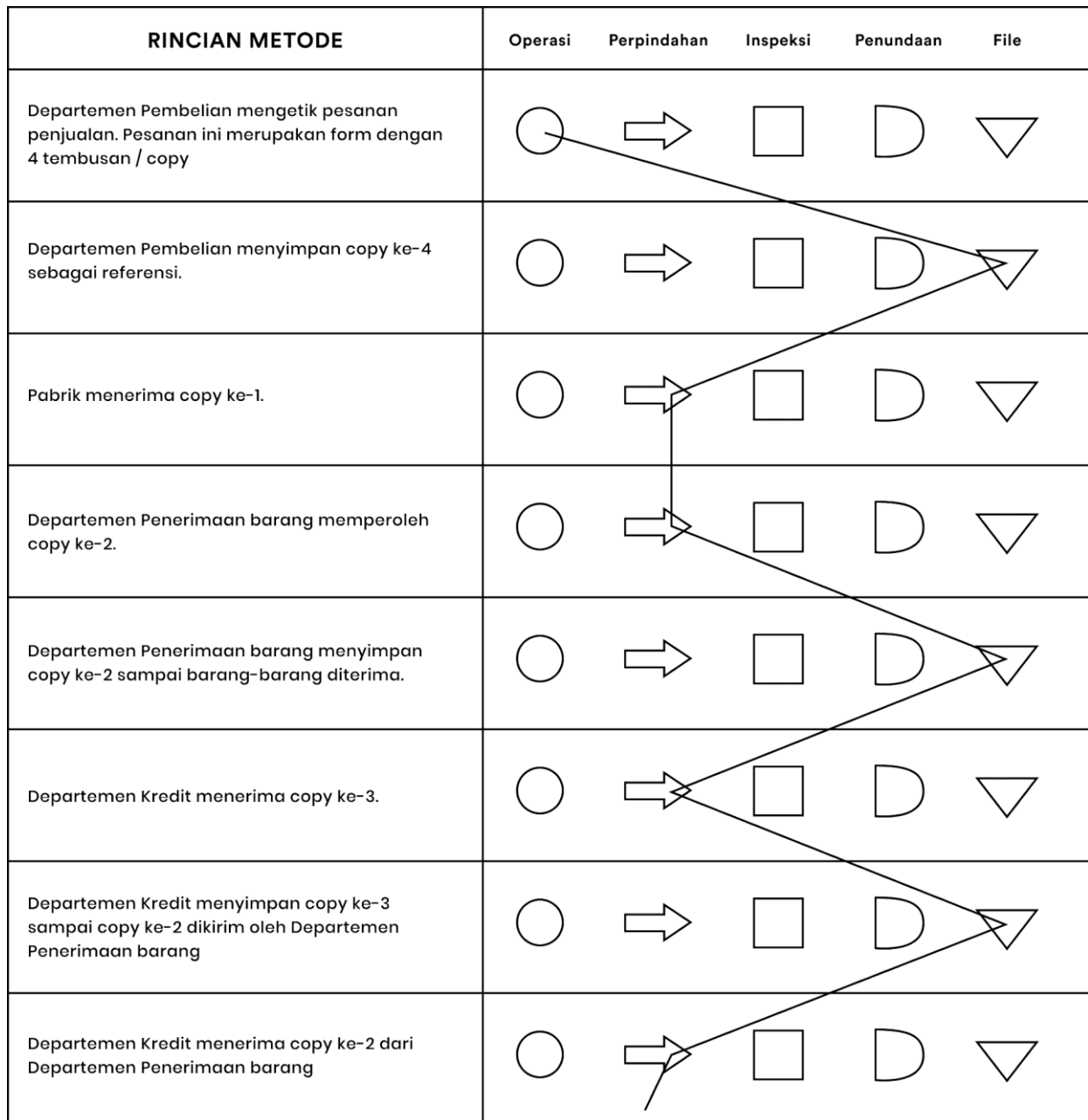
e. Flowchart Proses (Process Flowchart)

Flowchart Proses adalah Deskripsi teknik dari teknik industri terdesentralisasi dan analisis langkah-langkah selanjutnya dalam proses atau sistem. Diagram alir memiliki lima simbol khusus, misalnya pada Gambar 18.



Gambar 18. Simbol-Simbol Flowchart Proses

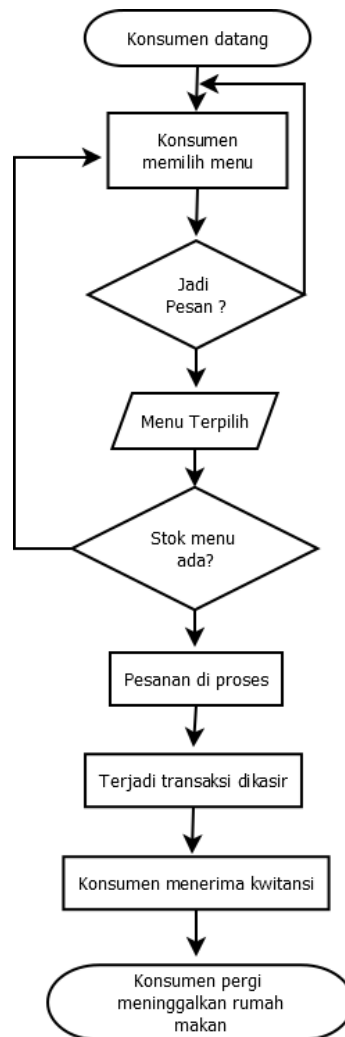
Flowchart Proses biasanya digunakan oleh insinyur industri untuk meneliti dan mengembangkan proses manufaktur. Dalam analisis sistem, diagram alir ini dapat digunakan secara efektif untuk melacak arus laporan atau formulir. Pada Gambar 19 merupakan ilustrasi contoh diagram alir proses.



Gambar 19. Flowchart Proses

C. SOAL LATIHAN/TUGAS

1. Apa itu flowchart menurut pendapatmu!
2. Gambarkan flowchart dari rumus persegi panjang!
3. Buatlah flowchart kegiatan sehari-hari!
4. Jelaskan algoritma dari flowchart di bawah ini:



5. Mengapa seorang programmer harus bisa membaca flowchart?

D. REFERENSI

IBM. (1969). Flowcharting techniques. (C20-8152-1 ed.). New York: IBM, Technical Publications Department.

Tague, N. R. (2005). The quality toolbox. (2th ed.). Milwaukee, Wisconsin: ASQ Quality Press. Available from <http://asq.org/quality-press/displayitem/index.html?item=H1224>

Burch, J.G., System, Analysis, Design, and Implementation, Boyd & Fraser Publishing Company, 1992

Yourdon Edward, Modern Structur Analisis, Prentice – Hall, Inc, 1989.

Deutsches Institut für Normung. (September 1966). Sinnbilder für datenfluß- und programmablaufpläne. Deutsche Industrienorm DIN 66001. Tiergarten, Berlin: DIN.

GLOSARIUM

American National Standards Institute (ANSI) adalah sebuah lembaga nirlaba swasta yang mengawasi pengembangan standar konsensus sukarela untuk produk, jasa, proses, sistem, dan personel di Amerika Serikat.

ISO adalah singkatan dari *The International Organization for Standardization*, yaitu organisasi internasional untuk standardisasi yang menetapkan standar internasional dibidang industri dan komersial di dunia dimana ia bertujuan untuk meningkatkan perdagangan antar negara di dunia.

Form adalah sebuah objek kontrol penampung dari objek kontrol lain, dapat menerima tugas dan memberikan reaksi terhadap tindakan dari pengguna untuk kelangsungan sebuah program aplikasi.

Peripheral adalah hardware tambahan yang disambungkan ke komputer, biasanya dengan bantuan kabel ataupun sekarang sudah banyak perangkat peripheral wireless.

Manufaktur adalah suatu cabang industri yang mengoperasikan peralatan, mesin dan tenaga kerja dalam suatu medium proses untuk mengolah bahan baku, suku cadang, dan komponen lain untuk diproduksi menjadi barang jadi yang memiliki nilai jual.

PERTEMUAN 7

ORIENTASI OBJEK

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang konsep perancangan berorientasi objek, konteks sistem dan model penggunaan, serta perancangan arsitektural. Dari pertemuan diharapkan mahasiswa mampu memahami konsep pendekatan perancangan berorientasi objek.

B. URAIAN MATERI

1. Konsep Perancangan Berorientasi Objek

Perancangan berorientasi objek adalah teknik atau metode baru untuk menemukan masalah dan sistem (sistem perangkat lunak, sistem informasi atau sistem lain). Pendekatan berorientasi objek akan memperlakukan sistem yang akan dikembangkan sebagai kumpulan objek yang sesuai dengan objek dunia nyata. Konsep "berorientasi objek" berarti kita mengatur perangkat lunak sebagai kumpulan objek tertentu dengan struktur data dan perilaku.

a. Karakteristik Dari Objek

Dalam desain berorientasi objek, objek itu sendiri memiliki banyak karakteristik. Berikut beberapa ciri dari benda tersebut:

Objek ialah Identitas artinya data yang diukur memiliki nilai tertentu, yang dapat membedakan entitas yang disebut objek. Objek bisa spesifik, seperti file dalam sistem, atau konseptual, seperti strategi penjadwalan dalam multiprocessing di sistem operasi. Setiap benda memiliki atribut yang melekat pada identitasnya. Meskipun semua nilai atribut sama, dua objek bisa berbeda.

Kelas Objek ialah Tinjauan sekumpulan objek yang dibagi menjadi atribut, operasi, metode, hubungan, dan arti yang sama. Mengumpulkan data (atribut) dan perilaku (operasi) dengan struktur data yang sama ke dalam sekumpulan aktivitas. Kelas objek adalah wadah untuk objek. Dapat

digunakan untuk membuat objek. Objek mewakili deskripsi fakta / kelas.

b. Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai 3 karakteristik utama:

Encapsulation merupakan Batasi cakupan program pada basis data yang diproses. Data dan prosedur atau fungsi dikemas dalam suatu objek sehingga prosedur atau fungsi eksternal lainnya tidak dapat mengaksesnya. Data tidak dilindungi oleh proses atau objek lain, kecuali proses dalam objek itu sendiri.

Inheritance adalah Ditunjukkan bahwa anak-anak dari objek akan secara langsung mewarisi data / atribut dan metode dari induknya. Properti dan metode objek di objek induk akan diteruskan ke objek turunan, dan seterusnya. Pewarisan berarti bahwa atribut dan operasi yang dibagi antar kelas memiliki hubungan hierarki. Anda biasanya dapat mengidentifikasi kelas dan kemudian menetapkan sebagai subkelas. Setiap subclass memiliki hubungan atau mewarisi semua properti yang dimiliki oleh kelas induknya, dan menambahkan properti uniknya. Properti dan layanan kelas objek dapat ditentukan dari kelas Objek lainnya. Pewarisan menggambarkan generalisasi kelas.

Polimorfisme yaitu tunjukkan konsep bahwa benda yang sama dapat memiliki bentuk dan perilaku yang berbeda. Polimorfisme berarti bahwa operasi yang sama mungkin berbeda dalam kategori yang berbeda. Kemampuan objek yang berbeda untuk melakukan metode yang sesuai dalam menanggapi pesan yang sama. Pilihan metode yang sesuai bergantung pada kelas di mana Objek harus dibuat.

c. Kelebihan dan Kekurangan Perancangan Berorientasi Objek

Berikut kelebihan dan kekurangan perancangan berorientasi objek sebagai berikut :

1) Kelebihan:

- a) Dibandingkan dengan metode SSAD, OOAD lebih mudah digunakan dalam konstruksi sistem.
- b) Tidak ada pemisahan antara tahap desain dan analisis, sehingga komunikasi antara pengguna dan pengembang ditingkatkan dari

awal hingga akhir pengembangan sistem..

- c) Analisis dan pemrograman tidak dibatasi oleh batasan implementasi sistem, sehingga mereka dapat mengonfirmasi desain melalui berbagai lingkungan eksekusi.
- d) Encapsulation data dan method agar dapat digunakan kembali dalam proyek lain, yang akan memudahkan proses desain, pemrograman, dan pengurangan harga.
- e) OOAD memungkinkan standarisasi objek, yang akan memudahkan pemahaman desain dan mengurangi risiko implementasi proyek.
- f) Dekomposisi obyek, Memungkinkan analisis untuk menguraikan masalah menjadi bagian-bagian masalah dan bagian-bagian yang dikelola secara terpisah. Kode program dapat bekerja sama. Metode ini memungkinkan pengembangan perangkat lunak yang cepat, sehingga dapat segera dicantumkan dan kompetitif. Sistem akhir sangat fleksibel dan mudah dirawat.

2) Kekurangan:

- a) Pada tahap awal desain OOAD, sistemnya mungkin sangat sederhana.
- b) OOAD lebih fokus pada pengkodean daripada SSAD.
- c) OOAD tidak menekankan kinerja tim seperti SSAD.
- d) Di OOAD, tidak mudah untuk mendefinisikan kelas dan objek yang dibutuhkan oleh sistem.
- e) Umumnya, pemrograman berorientasi objek digunakan untuk menganalisis fungsi sistem, dan metode OOAD tidak didasarkan pada fungsi sistem.

2. Konteks Sistem dan Model Penggunaan

Pemodelan berorientasi objek dibagi menjadi dua bentuk, yaitu sebagai berikut:

a. Pemodelan Sebagai Teknik Desain

Teknik pemodelan objek menggunakan tiga model untuk menggambarkan sistem:

1) Model Objek

- a) Model objek menggambarkan struktur statis dan hubungan objek

dalam sistem.

- b) Model objek berisi diagram objek. Grafik objek adalah grafik di mana node adalah kelas-kelas yang memiliki hubungan antar kelas.

2) Model Dinamik

- a) Model dinamik menggambarkan aspek sistem yang berubah seiring waktu.
- b) Model dinamik Digunakan untuk mengekspresikan aspek kontrol dari sistem.
- c) Model dinamik terdapat state diagram. State diagram merupakan graph yang nodenya adalah state dan arc adalah pergatian antara state yang disebabkan oleh event.

3) Model Fungsional

- a) Model fungsional Menjelaskan konversi nilai data dalam sistem.
- b) Model fungsional Berisi diagram aliran data. DFD adalah grafik, di mana node menggambarkan proses, dan busur adalah aliran data.

b. Model Berorientasi Objek

Model objek menangkap struktur statis sistem dengan mendeskripsikan objek-objek dalam sistem, hubungan antar objek, serta karakteristik dan atribut dari setiap kelas dan objek. Model berorientasi objek lebih mendekati keadaan sebenarnya, dan dilengkapi dengan representasi grafis dari sistem, yang sangat berguna untuk berkomunikasi dengan pengguna dan mendokumentasikan struktur sistem..

1) Objek

- a) Objek didefinisikan sebagai konsep, abstrak atau objek dengan batasan dan makna masalah.
- b) Semua benda memiliki identitas yang berbeda dengan benda lainnya. Istilah "identitas" berarti bahwa objek dibedakan berdasarkan atribut yang melekat daripada dengan deskripsi atributnya. Misalnya, meskipun kembar identik terlihat mirip, mereka adalah dua orang yang berbeda.
- c) Terkadang objek mewakili item, jadi istilah "instance objek" dan "kelas objek" digunakan untuk menunjukkan sekelompok item yang identik.

2) Kelas

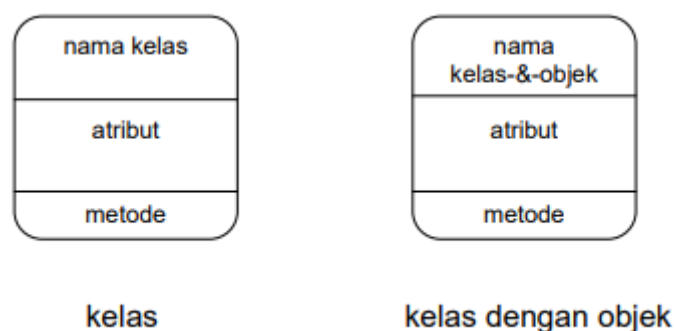
- a) Kelas objek mendeskripsikan sekumpulan objek dengan atribut (atribut), perilaku umum (operasi), hubungan umum dengan objek lain, dan semantik umum. Contoh: orang, perusahaan, hewan, proses adalah objek. Setiap orang memiliki usia, IQ, dan mungkin pekerjaan. Setiap proses memiliki pemilik, prioritas, dan daftar sumber daya yang diperlukan.
- b) Objek kelas dan objek biasanya sama dengan objek dalam deskripsi masalah.

3. Diagram Objek

Diagram objek adalah pelengkap simbol grafis dan digunakan untuk memodelkan objek, kelasnya, dan hubungannya dengan objek lain. Diagram objek berguna untuk pemodelan dan pemrograman abstrak.

a. Kelas dan Objek

Konsep dasar dari analisis berorientasi objek adalah objek itu sendiri. Objek adalah entitas yang berisi data dan metode. Kelas adalah satu atau lebih objek dengan properti dan metode yang sama, dan kelas - & - objek adalah kelas dengan satu atau lebih objek di dalamnya. Nama kelas adalah kata benda tunggal, atau kata sifat dan kata benda. Nama kelas - & - objek harus menggambarkan satu objek kelas.

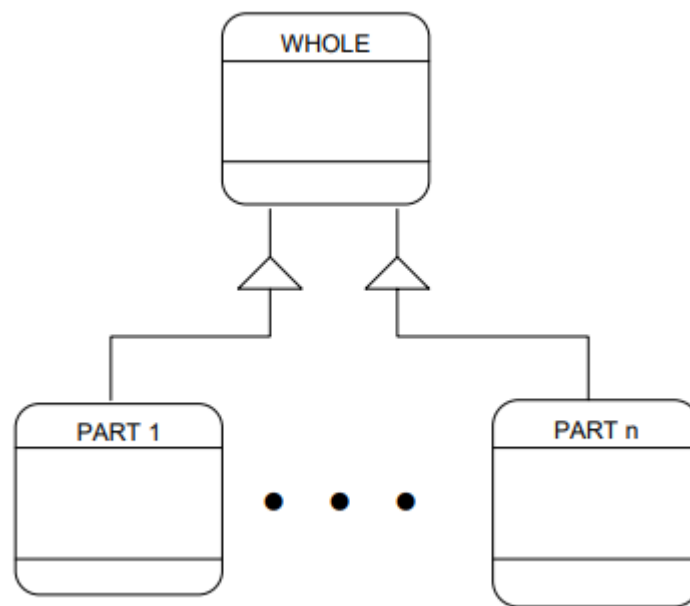


Gambar 20. Notasi untuk kelas dan kelas-&-objek

b. Struktur Objek dan Hirarki Kelas

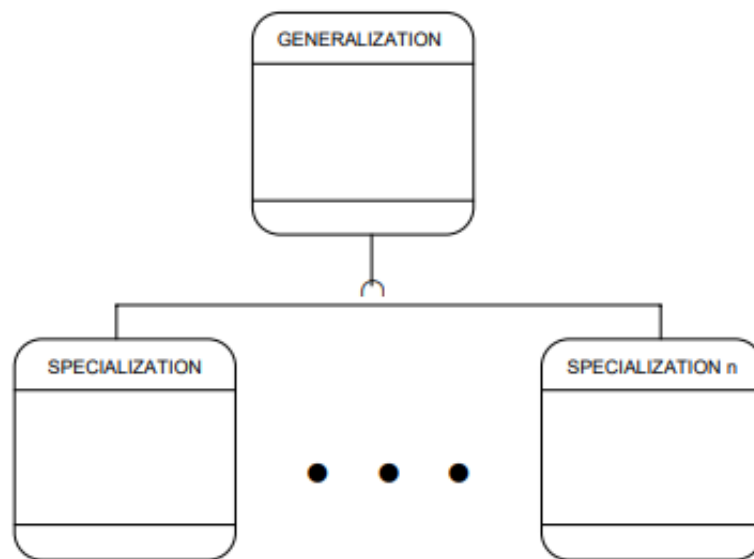
Struktur kelas dibagi menjadi dua jenis, yaitu **Whole-Part Structure** dan **Gen-Spec Structure**.

Whole-Part Structure Menampilkan struktur hierarki dari satu kelas sebagai komponen kelas lain. Kelas ini juga disebut sub-objek. Misalnya, kelas Mobil adalah Utuh, dan komponennya Mesin, Bingkai, dll. Adalah Part1, Part 2, ..., Partn.



Gambar 21. Notasi untuk whole-part structure

Gen-Spec Structure Tunjukkan kelas tersebut sebagai kursus khusus dari kelas di atas. Kelas dengan karakteristik umum disebut Generalisasi, Superclass atau Topclass, dan kelas dengan atribut khusus disebut Spesialisasi.



Gambar 22. Notasi untuk gen-spec structure

Misalnya, kategori "mobil" adalah "tujuan umum", sedangkan "mobil", "truk", "mobil penumpang kecil", dll. Adalah "spesialisasi 1", "spesialisasi 2", ..., "spesialisasi". Kategori fitur.

Atribut

Data yang dideskripsikan oleh atribut dapat memberikan informasi tentang kelas atau objek di mana atribut tersebut berada.

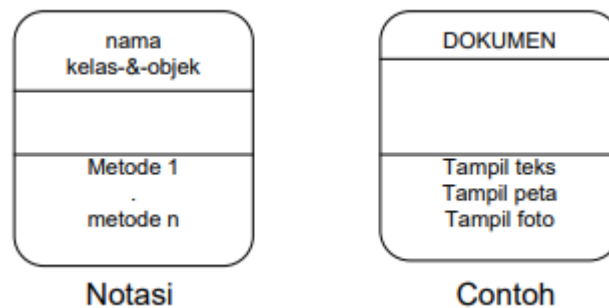


Gambar 23. Notasi untuk atribut

Metode

Metode (metode), juga disebut layanan atau operator, adalah prosedur atau fungsi yang biasanya ditemukan dalam bahasa Pascal, tetapi cara kerjanya sedikit berbeda. Metode adalah subrutin, yang menggabungkan objek dan

atribut. Metode ini digunakan untuk mengakses data yang terdapat dalam objek tersebut.



Gambar 24. Notasi untuk metode

Pesan (Message)

Message ini adalah metode menghubungkan satu objek dengan objek lainnya. Pesan yang dikirim oleh objek ke objek tertentu dapat dijelaskan dengan panah.



Gambar 25. Notasi untuk message

4. Perancangan Arsitektural

Arsitektur bias bergantung pada sudut pandang tertentu, itu bisa berarti berbagai hal. Dari perspektif PBO, arsitektur standar UML merupakan aspek integral dari organisasi sistem. Aspek-aspek tersebut merupakan bagian dari arsitektur sistem, yang terdiri dari fungsi, struktur dan proses (fungsi, statik dan perilaku). Selain itu, arsitektur juga mengatasi kendala kinerja, skalabilitas, penggunaan kembali, serta ekonomi dan teknis..

Secara garis besar, arsitektur dapat dilihat dari sisi eksternal (pengguna), yaitu dalam bentuk fungsional. Selain itu, Anda juga dapat melihat bentuk

properti, metode, dan kelas dari dalam. Arsitektur juga melibatkan hubungan antar objek, objek ini berisi pesan untuk menjalankan fungsi tertentu.

Setelah interaksi antara sistem perangkat lunak yang dirancang dan lingkungan sistem didefinisikan sehingga dapat menggunakan informasi ini sebagai dasar untuk merancang arsitektur sistem..

- a. Lapisan antarmuka, tangani semua komunikasi eksternal.
- b. Lapisan pengumpulan data mengumpulkan dan merangkum semua data yang dibutuhkan oleh sistem
- c. Lapisan instrumen, merangkum semua instrumen untuk mengumpulkan data mentah

Ada dua desain arsitektur, sebagai berikut:

a. Arsitektural Logika

- 1) Arsitektur database yang menyediakan entitas dan hubungan tingkat rendah
- 2) Lapisan kedua meliputi proses bisnis inti dan logika bisnis yang diproses oleh sistem
- 3) Lapisan ketiga menyiapkan informasi rinci tentang aplikasi yang mendukung berbagai fungsi bisnis yang dibangun dari sistem ERP
- 4) Pengguna akhir:
 - a) Tingkat pertama dan kedua tidak pernah terlihat, karena mereka biasanya berinteraksi di tingkat aplikasi untuk menyediakan akses ke aplikasi fungsional
 - b) Hanya mengakses lapisan ketiga (aplikasi antarmuka klien-pengguna). Mereka memberikan layanan kepada pengguna akhir dalam bentuk akses fungsional, termasuk setiap fungsi dan modul yang dicakup oleh sistem ERP.

b. Arsitektural Fisik

- 1) Fokus pada efisiensi sumber daya sistem.
- 2) Sumber daya yang terlibat: biaya, waktu, respons terhadap jumlah perangkat, dll..
- 3) Jadikan seluruh sistem lebih skalabel dan kurangi sumber daya yang diperlukan.

C. SOAL LATIHAN/TUGAS

1. Sebutkan dan jelaskan model pada orientasi objek!
2. Sebutkan dan jelaskan Karakteristik pada Objek!
3. Menurut anda apa keuntungan dan kekurangan menggunakan orientasi objek!
4. Sebutkan dan jelaskan metodologi berorientasi Objek!
5. Jelaskan yang dimaksud Whole-Part Structure dan Gen-Spec Structure!

D. REFERESI

Rumbaugh James, et all, 1999, "The Unified Modeling Language Reference Manual".

Hoffer, A.Jeffrey and George, F, Joey, *Modern System Analysis and Design*, Prentice Hall-Inc, 2002

McGraw Hill., *UML*, 1999

Edward V. Berard, *Origins Objects Oriented Technology*

GLOSARIUM

User adalah pengguna pada layanan atau perangkat dalam sistem teknologi informasi.

Methods atau metode adalah kumpulan pernyataan yang dikelompokkan bersama-sama untuk melakukan operasi.

Class atau Kelas adalah cetak biru untuk membuat objek.

Reuse berarti 'menggunakan kembali'. Langkah ini mengajak kita untuk menggunakan kembali produk yang sudah dipakai.

+Enkapsulasi (encapsulation) adalah sebuah metoda untuk mengatur struktur class dengan cara menyembunyikan alur kerja dari class tersebut.

PERTEMUAN 8

ORIENTASI OBJEK (LANJUTAN)

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian dan proses analisa sistem. Dari pertemuan ini diharapkan mahasiswa mampu mendeskripsikan apa itu analisa sistem.

B. URAIAN MATERI

1. Identifikasi Objek

Dalam Identifikasi objek ada beberapa metodologi berorientasi objek yang mempunyai 3 karakteristik utama:

a. Encapsulation (Pengkapsulan)

- 1) Enkapsulasi adalah dasar dari parameter lingkup program untuk data yang diolah.
- 2) Data dan proses atau fungsi disimpan bersama sebuah objek sehingga tidak ada prosedur atau fungsi lain yang diperkenalkan dari luar bisa mengunjunginya.
- 3) Lindungi data agar tidak terpengaruh oleh proses atau objek lain (kecuali proses) berada di dalam objek itu sendiri.

b. Inheritance (Pewarisan)

- 1) Pewarisan adalah teknik untuk menunjukkan anak suatu benda akan langsung mewarisi data / atribut dan proses dari induknya. Properti dan metode objek induk diteruskan ke Anak objek, dll.
- 2) Warisan berarti atribut dan operasi dimiliki antara kelas terkait Hirarki.
- 3) Suatu kelas dapat menentukan kategori terlebih dahulu, lalu menentukan dibagi secara khusus ke dalam sub-kategori. Setiap subclass memiliki hubungan atau mewarisi semua properti yang dimiliki oleh kelas induk, dan ditambah itu memiliki atribut unik. Objek kelas dapat mendefinisikan atribut dan layanan kelas objek lain.

- 4) Inheritance menjelaskan generalisasi suatu kelas

Contohnya:

- a) Mobil dan sepeda motor adalah subkategori kendaraan bermotor.
- b) Kedua subclass ini mewarisi atribut yang dimiliki oleh kendaraan bermotor, dengan mesin, dan bisa berjalan.
- c) Kedua subkategori ini memiliki karakteristiknya masing-masing beda, seperti jumlah roda dan kemampuan berjalan.

c. Polymorphism (Polimorfisme)

- 1) Polimorfisme adalah sebuah konsep, yang menunjukkan bahwa ada sesuatu orang yang sama dapat memiliki bentuk dan perilaku yang berbeda.
- 2) Polimorfisme berarti operasi yang sama dimungkinkan ada perbedaan antara kelas yang berbeda.
- 3) Kemampuan untuk mengeksekusi metode untuk objek yang berbeda tanggapan yang tepat untuk pesan yang sama.
- 4) Pilih metode yang sesuai menurut kategori mana objek harus dibuat.

2. Model Desain

OOAD yang dikemas mencakup penggunaan metode objek untuk analisis dan desain sistem. Object-Oriented Analysis (OOA) adalah metode analisis yang meneliti persyaratan (persyaratan / persyaratan yang harus dipenuhi oleh sistem) dari perspektif kelas dan objek yang ditemui di dalam perusahaan. Pada saat yang sama, desain berorientasi objek (OOD) adalah metode memandu arsitektur perangkat lunak berdasarkan operasi objek sistem atau subsistem. Ada beberapa konsep dasar di OOAD, yaitu :

a. Objek (object)

Objek adalah objek fisik dan konseptual di sekitar kita. Beberapa contoh objek, seperti perangkat keras, perangkat lunak, dokumen, orang, konsep, dll. Misalnya, untuk tujuan pemodelan, eksekutif memperlakukan karyawan, gedung, departemen, arsip, dan keuntungan perusahaan sebagai objek. Pada saat yang sama, teknisi otomotif akan mempelajari ban, pintu, mesin, kecepatan dan nomor tertentu. Bahan bakar adalah obyeknya. Contoh lain adalah insinyur perangkat lunak memperlakukan tumpukan,

antrian instruksi, jendela, dan kotak centang sebagai objek.

State dari sebuah objek adalah kondisi dari objek atau sekumpulan status yang menggambarkan objek tersebut. Misalnya, status rekening tabungan dapat mencakup saldo saat ini, status jam tangan adalah rekor saat ini, dan status bola lampu adalah "hidup" atau "mati". Status diwakili oleh nilai properti objek.

Atribut adalah Nilai internal objek, yang mencerminkan karakteristik objek, kondisi sesaatnya, hubungannya dengan objek lain, dan identifikasinya. Perubahan status tercermin melalui perilaku objek.

Behaviour atau Perilaku suatu objek mendefinisikan perilakunya (perilaku) dan bagaimana reaksinya. Perilaku ditentukan oleh rangkaian semua atau lebih operasi yang dapat dilakukan oleh objek itu sendiri. Perilaku suatu objek direfleksikan oleh antarmuka, layanan, dan metodenya.

Interface adalah Akses ke pintu layanan.

Service adalah fungsi yang dapat dilakukan oleh objek.

Method adalah mekanisme internal dari objek yang mencerminkan perilaku dari objek tersebut.

b. Kelas (Class)

Class adalah definisi umum sekelompok objek serupa (pola, templat, atau cetak biru). Kelas menentukan perilaku (perilaku) dan properti objek. Kelas adalah abstraksi entitas di dunia nyata. Dan objek adalah contoh kelas.

Misalnya, atribut hewan adalah hewan berkaki empat dan memiliki ekor. Tingkah lakunya adalah makan dan tidur. Sedangkan hewan contoh adalah kucing, gajah dan kuda.

c. Kotak Hitam (Black Boxes)

Sebuah Objek adalah kotak hitam. Konsep ini menjadi dasar realisasi objek. Dalam operasi OO, hanya pengembang (pemrogram, desainer, analis) yang dapat memahami detail proses yang terdapat dalam kotak hitam, dan pengguna tidak perlu mengetahui operasi yang sedang dilakukan, tetapi yang terpenting adalah mereka dapat menggunakan objek untuk memproses mereka Permintaan.

Encapsulation, Proses menyembunyikan detail implementasi suatu objek. Satu-satunya cara untuk mengakses data objek adalah melalui antarmuka.

Antarmuka ini dapat melindungi keadaan internal objek dari "gangguan" eksternal. Oleh karena itu, objek direpresentasikan sebagai kotak hitam untuk menerima dan mengirim pesan. Dalam OOP, kotak hitam berisi kode (instruksi yang dapat dimengerti komputer) dan data (informasi yang memerintahkan operasi padanya). Dalam OOP, kode dan data ditempatkan di "objek", dan konten objek disembunyikan, yaitu objek. Pengguna subjek tidak perlu mengetahui isi kotak. Untuk berkomunikasi dengan objek tersebut, diperlukan sebuah pesan.

Message adalah Proses menyembunyikan detail implementasi suatu objek. Satu-satunya cara untuk mengakses data objek adalah melalui antarmuka. Antarmuka ini dapat melindungi keadaan internal objek dari "gangguan" eksternal. Oleh karena itu, objek direpresentasikan sebagai kotak hitam untuk menerima dan mengirim pesan. Dalam OOP, kotak hitam berisi kode (instruksi yang dapat dimengerti komputer) dan data (informasi yang memerintahkan operasi padanya). Dalam OOP, kode dan data ditempatkan di "objek", dan konten objek disembunyikan, yaitu objek. Pengguna subjek tidak perlu mengetahui isi kotak. Untuk berkomunikasi dengan objek tersebut, diperlukan sebuah pesan.

d. Asosiasi dan Agregasi

Asosiasi adalah hubungan yang berarti antara banyak objek. Asosiasi direpresentasikan dengan menghubungkan garis antar objek. Contoh: asosiasi antara objek mobil dan seseorang. Sebuah mobil dapat dimiliki oleh satu atau beberapa orang, dan satu orang dapat memiliki nol, satu atau lebih mobil

Asosiasi antara Karyawan tempat kerja. Seorang karyawan bekerja di satu unit kerja. Sebuah unit kerja dapat memiliki beberapa karyawan.

Agregasi adalah Bentuk asosiasi khusus yang menjelaskan bahwa semua bagian suatu objek adalah bagian dari objek lain. Contoh: Kopling dan piston adalah bagian dari mesin. Mesin, roda, dan bodi adalah bagian dari mobil. Tanggal, bulan dan tahun adalah bagian dari tanggal lahir. Padahal, tanggal lahir, nama, alamat, dan jenis kelamin menjadi bagian dari identitas seseorang.

3. Spesifikasi Interface Objek

Desain antarmuka adalah proses untuk menentukan bagaimana sistem berinteraksi dengan entitas eksternal (seperti pelanggan, pemasok, dan sistem lainnya). Tentukan bagaimana pengguna berinteraksi dengan sistem (bagaimana memberikan masukan dan mendapatkan keluaran, dan bagaimana sistem menerima dan menjalankan perintah).

a. User Interface Problems

Dalam pembuatan interface pada system, biasanya ada masalah-masalah yang dialami oleh user. Berikut ini beberapa masalah yang sering ditemui:

- 1) Penggunaan istilah komputer yang ekstensif.
- 2) Desain tidak jelas
- 3) Tidak dapat membedakan tindakan Pilih ("Apa yang harus saya lakukan lanjut").
- 4) Bukan solusi untuk masalah tersebut konsisten.
- 5) Desain tidak konsisten.

b. Prinsip Desain Interface

Untuk membuat desain interface yang nyaman saat digunakan, terdapat beberapa prinsip yang bisa dijadikan acuan. Dari buku "General Principles Of UI Design" ditulis oleh Deborah J. Mayhew, ada 17 prinsip desain antarmuka secara umum.

- 1) User Compatibility (Kompatibilitas Pengguna).
Artinya, karena pengguna yang berbeda mungkin memiliki persyaratan tampilan yang berbeda, tampilan disesuaikan dengan tipikal pengguna. Misalnya, jika aplikasi cocok untuk anak-anak, jangan gunakan bahasa atau tampilan dewasa.
- 2) Product Compatibility (Kompatibilitas Produk)
Produk aplikasi akhir juga harus sesuai untuk orang awam dan ahli serta memiliki tampilan yang sama / mirip.
- 3) Task Compatibility (Kompatibilitas Tugas)
Fungsi tugas yang ada harus sesuai dengan tampilannya. Misalnya untuk opsi laporan, user akan langsung menjelaskan laporan yang akan ditampilkan.
- 4) Work Flow Compatibility (Kompatibilitas Alur Kerja)

Aplikasi dapat dijalankan untuk berbagai pekerjaan dalam satu tampilan, dengan pertimbangan tidak terlalu kelebihan beban.

5) Consistency (Konsisten)

Desainnya harus konsisten. Misal: jika menggunakan kata "Save" artinya "Simpan" kemudian terus digunakan kata itu dan jangan sampai berubah..

6) Familiarity (Keakraban)

Faktor kebiasaan masih mudah diidentifikasi. Seperti Ikon floppy disk akan lebih familiar digunakan sebagai icon Simpan perintah.

7) Simplicity (Kesederhanaan)

Aplikasi harus mengumpulkan semua yang ada di dalamnya lebih sederhana dan lebih bermakna.

8) Direct Manipulation (Manipulasi Langsung)

Aktivitas tersebut disajikan langsung kepada pengguna (user) sehingga komputer melakukan aktivitas tersebut saat pengguna memberikan instruksi langsung pada layar komputer. Contoh: Untuk membuat tulisan menjadi tebal, tekan saja ctrl + B.

9) Control (Kontrol)

Kontrol penuh pengguna, pengguna biasa biasanya tidak membutuhkan terlalu banyak aturan

10) WYSIWYG (What You See Is What You Get)

Ada korespondensi satu-ke-satu antara informasi di layar dan informasi pada hasil cetak atau file. Buat agar terlihat seperti pengguna asli dan pastikan fungsinya berfungsi seperti yang diharapkan.

Contoh: Ketik huruf A, huruf A output dilayar, dan jika ada maka hasilnya juga huruf A.

11) Flexibility (Fleksibilitas)

Sistem memiliki kemampuan untuk mencapai tujuan dengan berbagai cara. Sistem dapat beradaptasi dengan keinginan pengguna (bukan pengguna yang harus beradaptasi dengan kerangka desain sistem)

12) Responsiveness (Responsivitas)

Konten yang ditampilkan harus responsif. Misalnya, harap tunggu 68% untuk tampilan layar ... atau bilah pemuatan, dll.

13) Invisible Technology (Teknologi Tak Terlihat)

Tidak masalah algoritma mana yang pengguna tahu untuk digunakan.

Misalnya, untuk mengurutkan pengguna, tidak perlu mengetahui algoritme yang digunakan oleh programmer (pengurutan maks, pengurutan gelembung, pengurutan cepat, dll.).

14) Robustness (Kekokohan)

Dapat mengakomodasi kesalahan pengguna tanpa menampilkan kesalahan, apalagi mogok. Misalnya, jika pengguna memasukkan format email yang salah, masalah dapat diselesaikan dan umpan balik diberikan.

15) Protection (Perlindungan)

Lindungi pengguna dari kesalahan umum. Misalnya dengan memberikan fungsi return atau undo.

16) Ease of Learning (Mudah Dipelajari)

Mudah untuk mempelajari aplikasi pemula (awam). Ini akan memotivasi pengguna untuk menggunakannya.

17) Ease of Use (Mudah Digunakan)

membuat sistem yang mudah digunakan untuk semua kategori pengguna (awam atau ahli).

4. Proses Desain Interface

Untuk membuat Interface ada beberapa langkah yang bisa digunakan agar lebih efektif. Berikut ini adalah beberapa langkahnya:

a. Menggunakan User Skenario

Berikut ini adalah contoh skenario untuk “Melihat Data Diri Mahasiswa”.

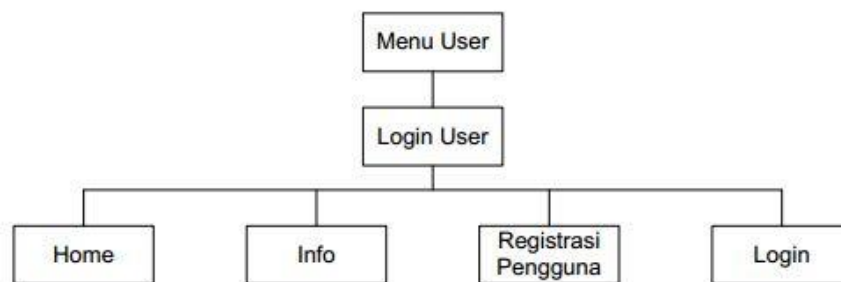
Use case Melihat Data Diri Mahasiswa	
Tujuan	Mengijinkan administrator untuk melihat data diri mahasiswa
Aktor	Administrator
Kondisi awal	Login tervalidasi dan valid
Skenario utama	<ol style="list-style-type: none"> 1. Administrator memilih menu data diri mahasiswa 2. Sistem menampilkan daftar mahasiswa, untuk dilihat data dirinya
Skenario alternatif	<ol style="list-style-type: none"> 1. Jika data diri mahasiswa yang dipilih masih kosong, maka sistem akan menunjukkan pesan “data diri masih kosong”. 2. Jika data diri mahasiswa yang dipilih masih belum lengkap, maka sistem akan menampilkan pesan “data diri masih belum lengkap”.
Kondisi akhir	Sistem menampilkan data diri mahasiswa sesuai dengan identitas mahasiswa yang dipilih.

Gambar 26. Contoh User Skenario

b. Membuat Struktur Desain Interface

Desain struktur antarmuka mendefinisikan komponen dasar antarmuka dan bagaimana mereka bekerja sama untuk menyediakan fungsi bagi pengguna.

Diagram Struktur Antarmuka (ISD) digunakan untuk menunjukkan bagaimana sistem terkait menggunakan semua layar, formulir dan laporan, dan bagaimana pengguna berpindah dari satu ke yang lain-lain.



Gambar 27. Contoh Struktur Desain Interface

c. Membuat Prototype

Berikut adalah contoh prototype interface Insert Buku sebuah program.

Buku

Kode Buku	:	<input type="text"/>
Judul Buku	:	<input type="text"/>
Jenis Buku	:	<input type="text"/> ▼
Penulis	:	<input type="text"/>
Penerbit	:	<input type="text"/>
Tahun Terbit	:	<input type="text"/>
Sinopsis	:	<input type="text"/>
ISBN /ISSN	:	<input type="text"/>

Gambar 28. Prototype Interface Insert Buku

d. Evaluasi

- 1) Evaluasi heuristik: perbandingan dengan prinsip dasar UI
- 2) Evaluasi latihan: bertemu dengan pengguna
- 3) Pengujian kegunaan

C. SOAL LATIHAN/TUGAS

1. Sebutkan dan jelaskan tahapan untuk membuat desain interface!
2. Sebutkan dan jelaskan metodologi berorientasi objek!
3. Buatlah desain prototype desain interface selain yang ada di modul!
4. Apa saja masalah yang sering dialami user interface?
5. Apa itu Black Box? Jelaskan dengan gaya Bahasa Anda!

D. REFERENSI

Grady Booch, 1991, Object-Oriented Analysis and Design with Application, Benjamin/Cummings.

Dennis, Alan., Barbara Halley Wixom and Roberta M. Roth. 2012. System Analysis and Design 5 th Edition. John Willey and Sons, Inc. New Jersey

Satzinger, John., Robert Jackson and Stephen Burd. 2010. System Analysis and Design in Changing World 5 th Edition. Cengage Learning. Boston.

GLOSARIUM

User Interface adalah Bagian visual dari situs web, aplikasi perangkat lunak atau perangkat keras, digunakan untuk menentukan bagaimana pengguna berinteraksi dengan aplikasi atau situs web dan bagaimana informasi ditampilkan di layar.

Prototype adalah proses perancangan sistem dengan membentuk contoh dan dimensi standar, dan Anda akan mengerjakannya nanti. Jika metode prototipe digunakan, pengembang dan pelanggan akan berinteraksi satu sama lain hingga diperoleh hasil terbaik.

Requirement adalah Kondisi atau kemampuan yang dibutuhkan oleh pengguna untuk memecahkan suatu masalah atau mencapai suatu tujuan

Heuristik adalah seni dan ilmu pengetahuan yang berhubungan dengan suatu penemuan.

Korespondensi adalah istilah umum yang merujuk kepada aktivitas penyampaian maksud melalui surat dari satu pihak kepada pihak lain.

PERTEMUAN 9

PENGENALAN UML

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini menjelaskan tentang pengenalan serta penjelasan UML, dan di sertakan penjelasan bagian – bagian dari UML.

B. URAIAN MATERI

1. Pengenalan UML

UML (Unified Modeling Language) adalah Bahasa pada memodelan atau pemodelan visual umum digunakan untuk menentukan, memvisualisasikan, membuat, dan merekam artefak dalam software. Ini menangkap keputusan dan pemahaman tentang sistem yang akan dibangun. Ini digunakan untuk memahami, merancang, mengeksplorasi, mengkonfigurasi, memelihara, dan mengontrol informasi tentang sistem. Ini dirancang untuk semua metode pengembangan, tahapan siklus hidup, domain aplikasi, dan media. Bahasa Pemodelan ini bertujuan untuk mengumpulkan pengalaman dalam teknik pemodelan masa lampau dan menggabungkan praktik terbaik perangkat lunak saat ini ke dalam metode standar.

UML mencakup simbol, konsep dan pedoman semantik. Ini memiliki bagian dinamis, statis lingkungan dan organisasi. Ini dirancang untuk didukung oleh alat pemodelan visual interaktif dengan generator kode dan penulis laporan. Spesifikasi UML tidak menentukan proses standar, tetapi dimaksudkan untuk digunakan dengan proses pengembangan berulang. Ini dirancang untuk mendukung sebagian besar proses perkembangan berorientasi objek saat ini.

UML memberikan informasi tentang struktur statis dan perilaku dinamis sistem. Sistem dimodelkan sebagai kumpulan objek individu yang berinteraksi untuk menyelesaikan pekerjaan dan pada akhirnya menguntungkan pengguna eksternal. Struktur statis mendefinisikan jenis objek yang penting bagi sistem dan implementasinya, serta hubungan antar objek. Perilaku dinamis mendefinisikan sejarah objek yang berubah seiring waktu dan komunikasi antar

objek untuk mencapai tujuan. Pemodelan sistem dari beberapa perspektif independen tetapi terkait dapat memahami berbagai tujuan sistem.

UML juga berisi struktur organisasi untuk mengatur model ke dalam paket perangkat lunak yang memungkinkan tim perangkat lunak untuk membagi sistem besar menjadi beberapa bagian yang dapat digunakan untuk memahami dan mengontrol ketergantungan antara paket perangkat lunak dan mengelola lingkungan pengembangan yang kompleks. Versi unit model. Ini berisi struktur yang digunakan untuk mewakili keputusan implementasi dan mengatur elemen runtime menjadi komponen.

UML bukanlah bahasa pemrograman. Alat dapat menyediakan generator kode dari UML ke berbagai bahasa pemrograman, dan juga dapat membangun model rekayasa terbalik dari program yang ada. UML bukanlah bahasa yang sangat formal untuk membuktikan teorema. Ada banyak bahasa seperti itu, tetapi untuk sebagian besar tujuan, bahasa tersebut tidak mudah dipahami atau digunakan. UML adalah bahasa pemodelan universal. Untuk bidang profesional seperti tata letak GUI, desain sirkuit VLSI, atau kecerdasan buatan berbasis aturan, alat yang lebih profesional menggunakan bahasa tertentu mungkin sesuai.

UML adalah bahasa pemodelan diskrit. Ini tidak dimaksudkan untuk memodelkan sistem berkelanjutan di bidang teknik dan fisika. UML bertujuan untuk menjadi bahasa pemodelan tujuan umum untuk sistem diskrit (misalnya, sistem yang dibuat oleh perangkat lunak, firmware, atau logika digital).

2. Pengertian UML

UML adalah bahasa yang digunakan untuk mendefinisikan, memvisualisasikan, membangun, dan merekam artefak (bagian dari informasi yang digunakan atau dihasilkan dalam proses pemodelan perangkat lunak, artefak ini dapat berupa model, deskripsi, atau perangkat lunak), seperti pemodelan bisnis dan alat non-sistem, dll. . . Selain itu, UML merupakan bahasa pemodelan yang menggunakan konsep berorientasi objek. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera Rational Software Corp.

Notasi yang diberikan oleh UML dapat membantu Anda memodelkan sistem dari berbagai sudut. UML tidak hanya digunakan untuk pemodelan perangkat lunak, tetapi UML digunakan di hampir semua area yang membutuhkan pemodelan

3. Bagian - bagian dari UML

Bagian utama dari UML adalah view, diagram, elemen model dan mekanisme umum.

a. View

View di pergunakan untuk melihat sistem model dari berbagai aspek. Tampilannya bukanlah grafik, tetapi abstraksi yang mengandung diagram.

Beberapa jenis view di UML meliputi: use case view, logical view, component

Use case view

Menjelaskan fungsi yang akan digunakan oleh peserta eksternal. Peserta yang berinteraksi dengan sistem dapat berupa pengguna atau sistem lain, dan pandangan tersebut dijelaskan dalam bentuk diagram use case atau diagram aktivitas..

View ini terutama digunakan oleh pelanggan, desainer, pengembang, dan penguji sistem.

Logical view

Mengdeskripsikan bagaimana fungsi Sistem, pola statis (class, object, dan relationship), dan kerja sama dinamis kelahirannya saat selaur korban mengangkat suruhan ke korban lain tambah kelebihan tertentu.

View tersebut dideskripsikan sebagai struktur statis berupa diagram class, dan model dinamis berupa diagram state, sequence, collaboration, dan activity diagram. View ini untuk digunakan oleh desainer dan pengembang.

Componen View

Implementasi dan dependensi modul. Komponen sebagai jenis modul kode lainnya mewakili struktur dan ketergantungannya, serta alokasi sumber daya komponen dan informasi manajemen lainnya.

Vew ini dideskripsikan sebagai tampilan komponen untuk digunakan

pengembang (developer).

Concurrency view

Pembagian sistem menjadi proses dan prosesor. View ini diwakili oleh skema dinamis (state, sequence, collaboration dan activity skema) dan skema implementasi (component dan deployment skema), dan digunakan oleh pengembang, integrator, dan penguji.

Deployment view

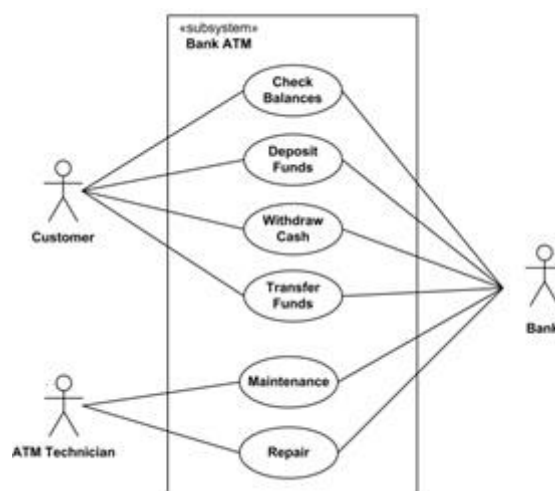
Mendekripsikan fisik sistem sebagai komputer dan perangkat (nodes) dan sebagai apa hubungannya dengan sistem lain. View ini dijelaskan dalam deployment diagram dan digunakan oleh pengembang, integrator, dan penguji.

b. Diagram

Diagram grafik menunjukkan simbol elemen model, yang disusun untuk menggambarkan bagian atau aspek tertentu dari sistem. Grafik adalah bagian dari tampilan tertentu, dan biasanya ditetapkan ke view tertentu saat menggambar. Jenis diagram antara lain :

Use Case Diagram

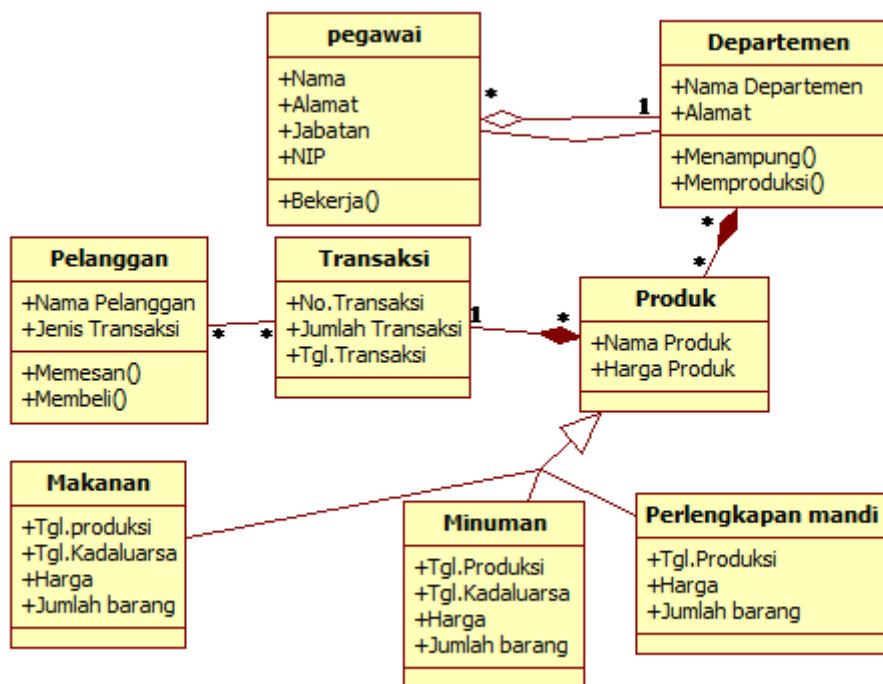
Menggambarkan sejumlah actor external dan hubungannya pakai use case yang diberikan oleh sistem. Use case adalah deskripsi fungsional yang disediakan oleh sistem bagian dalam bentuk teks sebagai dokumen simbol use case, tetapi juga bisa dilengkapi dalam activity diagram.



Gambar 9.3.2.1 Contoh Use Case Diagram

Class Diagram

Jelaskan struktur kelas statis internal sistem. Kelas mewakili item yang ditangani oleh sistem. Kelas dapat dihubungkan satu sama lain dalam berbagai cara: asosiasi (terhubung satu sama lain), ketergantungan (kelas bergantung pada/menggunakan keluarga lain), khusus (satu keluarga adalah keluarga lain) atau paket (digabungkan menjadi satu unit)) Sebuah sistem biasanya memiliki beberapa diagram kelas.

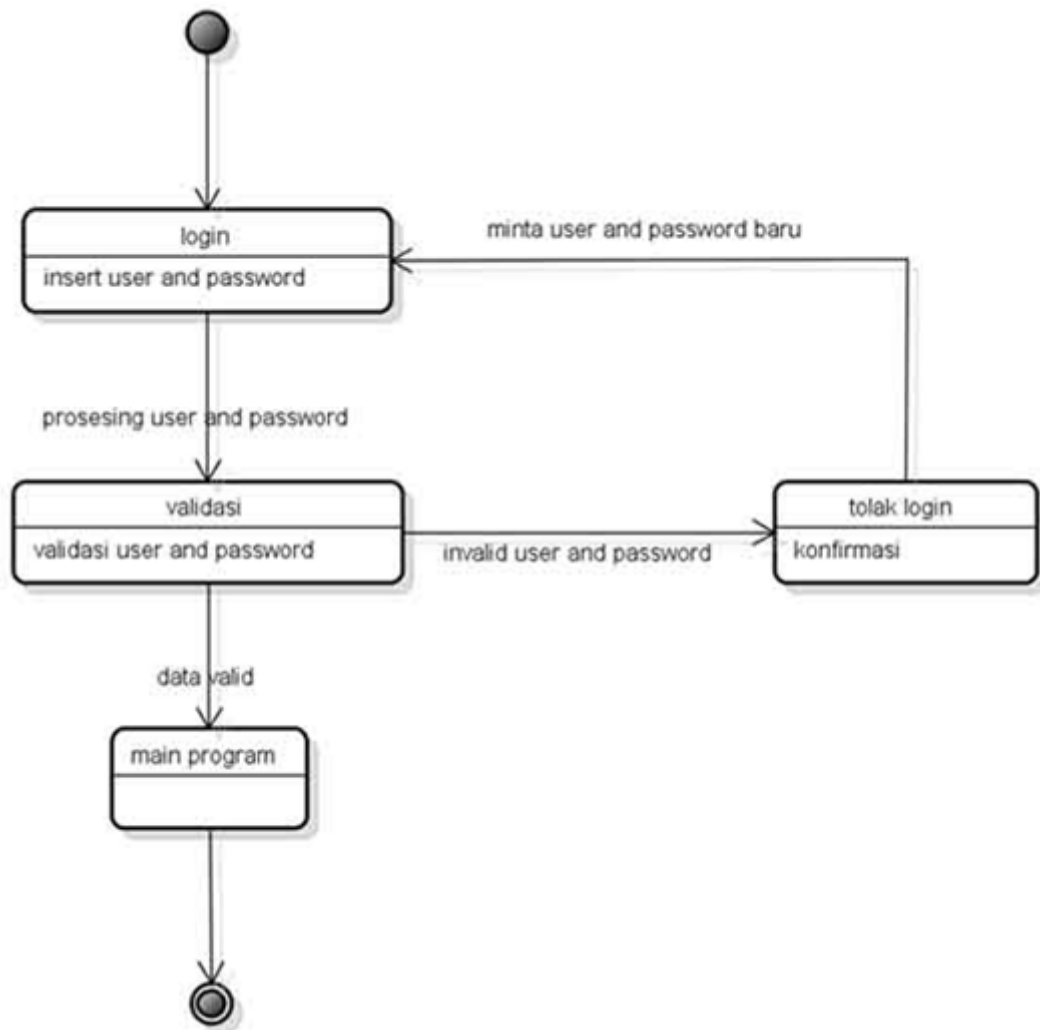


Gambar 9.3.2.2 Contoh Class Diagram

State Diagram

Menggambarkan semua state (kondisi) yang dimiliki oleh objek class dan ihwal yang menyebabkan suasana berubah. Acara bisa berupa objek lain yang mengirimkan pesan.

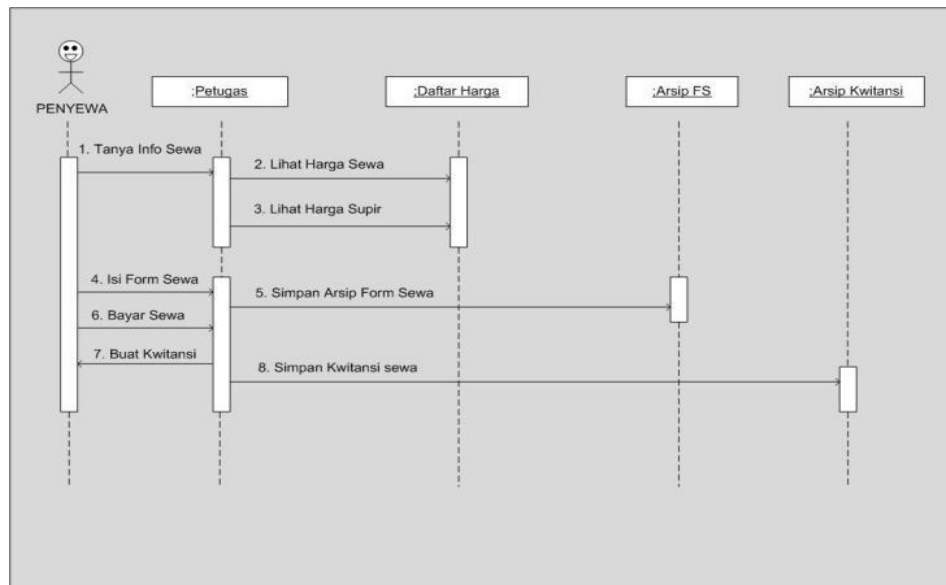
State class tidak di gambarkan semua class, semata-mata yang memegang beberapa state yang terdefinisi pakai kesetiaan dan mengenai class merangkak oleh state yang berbeda.



Gambar 9.3.2.3 Contoh State Diagram

1) Sequence Diagram

Menggambarkan kolaborasi dinamis sela banyak objek. Tujuannya adalah kepada mempresentasikan alur instruksi yang dikirim antar tujuan dan koneksi antar sasaran, yang kelahirannya pada titik tertentu bagian dalam eksekusi sistem.

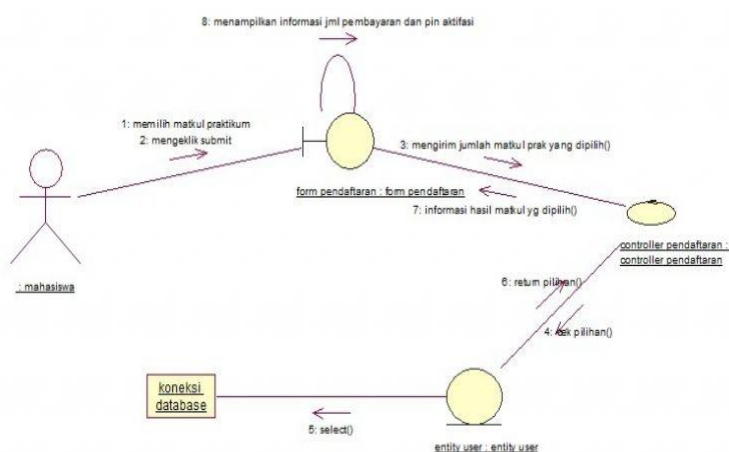


Gambar 9.3.2.4 Contoh Sequence Diagram

2) Collaboration Diagram

Menggambarkan kolaborasi dinamis, serupa rancangan urutan. Saat mempresentasikan pergantian pesan, collaboration rangka menceritakan tujuan dan hubungannya (mengacu ambang konteks).

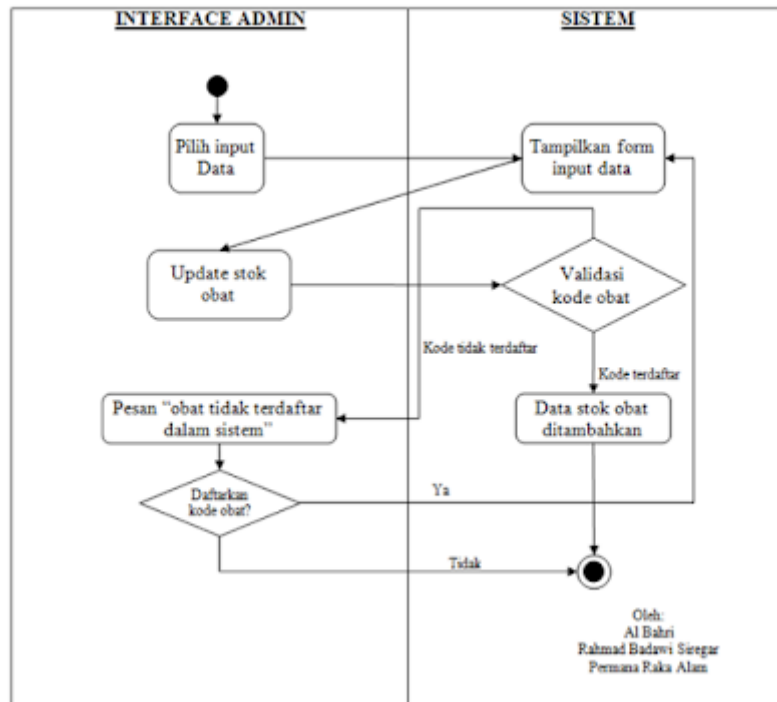
Jika fokus berada pada waktu atau urutan, gunakansequence diagram, tetapi jika fokus pada konteks,Gunakan collaboration diagram.



Gambar 9.3.2.5 Contoh Collaboration Diagram

3) Activity Diagram

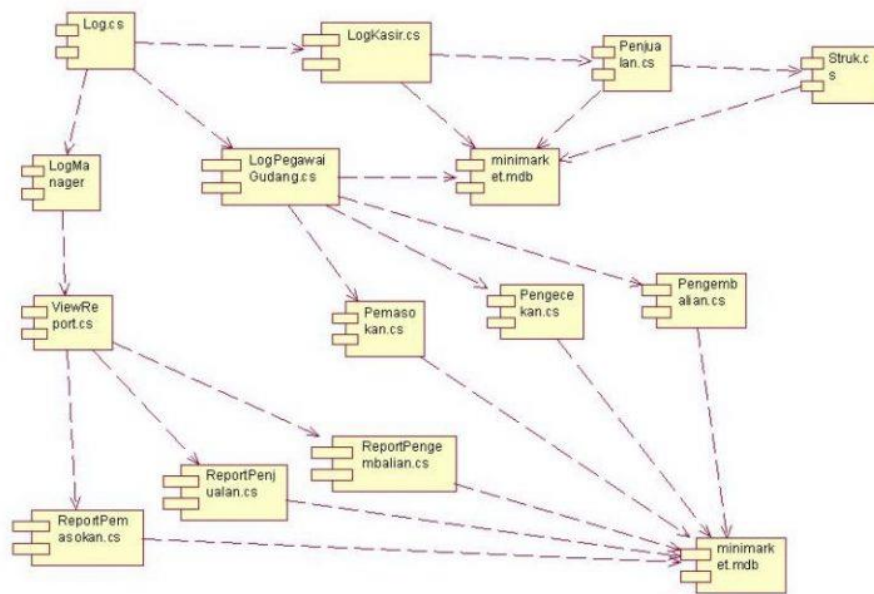
Mendesksripsikan rangkaian proses dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang terbentuk dalam operasi, sehingga dapat juga digunakan untuk aktivitas lain, seperti use case atau interaksi



Gambar 9.3.2.1 Contoh Activity Diagram

4) Componen Diagram

Menggambarkan struktur kode fisik komponen. Komponen bisa bercorak source code, elemen biner, atau executable. Komponen mengandung logic class atau class yang diimplementasikan kepada menegakkan pemetaan berasal view logical ke component view.



Gambar 9.3.2.7 Contoh Component Diagram

4. Gambaran dari UML

a. UML sebagai Bahasa Pemodelan

UML adalah bahasa pemodelan, mempunyai kosakata, dan menjadikan cara kepada mengeluarkan kekhawatiran mengenai konsep sistem dan fisika. Contoh sistem software intensif tekanan suara menunjukkan pandangan yang berbeda bersumber arsitektur sistem, mirip tambah siklus kehidupan pengembangan perangkat lunak kompilasi / pengembangan. Menggunakan UML akan memberi mengerti Anda cara membuat dan mempersembahkan formulir model yang baik, tetapi UML tidak bisa memasukkan mengerti kaca apa yang akan dibangun dan kapan harus bermanfaat kaca. Ini adalah institusi bagian dalam muslihat pengembangan perangkat lunak.

b. UML untuk Mengambarkan system

UML tidak hanya berupa rangkaian simbol grafik, tetapi setiap simbol bagian dalam pesan UML mengadakan distribusi semantik yang baik. Dengan hukum ini, satu dam dapat menggambar arketipe UML, sementara dam lain atau alat serupa lainnya bisa menafsirkan arketipe dengan jelas. Ini akan mengurangi kekhilafan yang disebabkan oleh friksi irama bagian dalam

persinggungan antara arketipe konseptual dan model lainnya.

UML menggambarkan model yang dapat dipahami dan disajikan dalam model teks suatu bahasa pemrograman. Anda dapat menebak contoh dalam model sistem berbasis web, tetapi Anda tidak dapat langsung memproses contoh tersebut dengan mempelajari kode sistem. Dengan menggunakan model UML, Anda dapat menggunakan bahasa pemrograman untuk membuat model dan merepresentasikan sistem Web.

c. UML untuk Menspesifikasikan system

Ini berarti membuat model yang mirip, eksplisit Dan lengkap. faktanya, uml mewakili seluruh pembahasan uraian penting, desain, dan perintah penjelmaan yang harus di buat tatkala peluasan dan operasi software intensif.

d. UML untuk membangun system

UML tidak suatu intonasi pemograman visual, tapi untuk pemodelan UML dapat di sambungkan secara langsung menjelang bahasa pemograman visual.

Bermaksud untuk berguna sama anutan yang bisa dimapping ke tekanan pemograman sebagai java, VB, C++ ataupun table ambang database relational atau pemilihan tetap depan database berorientasi object.

e. UML untuk pendokumentasikan system

Dengan kata lain, UML menunjukkan dokumen dan semua detail arsitektur sistem, uml menyediakan bahasa menampilkan permintaan dan tes. uml menyediakan bahasa untuk kegiatan pemodelan dalam rencana proyek dan mnejemen rilis.

5. Area Penggunaan UML

UML di Gunakan paling efektif pada domain seperti :

- a. System Informasi Perusahaan
- b. System Pernagkan , Perekonomian
- c. System bidang telekomuniikasi
- d. System bidang transportasi

- e. System bidang penerbangan
- f. System bidang perdagangan
- g. System bidang pelayanan elektronik
- h. System bidang pengetahuan
- i. System bidang pelayanan berbasis web

Tetapi uml tidak terbatas bagi pemodelan software pada faktanya uml banyak untuk pemodelan system non software seperti :

- a. Bagian dari kerja pada system perundangan
- b. Bagian struktirdan juga kekuatan dari sebuah system kepedulian kesehatan pasien
- c. Bagian dari desain hardware

6. Komponen – komponen yang terlibat dalam Use Case Diagram

a. Actor

Dengan kata lain, UML menunjukkan dokumen dan semua detail arsitektur sistem, uml juga menyediakan bahasa untuk menampilkan permintaan dan tes. UML menyediakan bahasa untuk kegiatan pemodelan dalam perencanaan proyek dan manajemen rilis..

Ada banyak kemungkinan yang menyebabkan para partisipan ini terkait dengan sistem, diantaranya :

- 1) Bagi yang tertarik dengan sistem, ada arus informasi dan arus informasi yang masuk ke dalam sistem
- 2) Orang atau kelompok yang akan mengelola sistem.
- 3) Sumber daya eksternal yang digunakan oleh sistem.
- 4) Sistem lain yang berinteraksi dengan sistem yang akan dibuat.

b. Use Case

Use Case merupakan bentuk dari fungsi sistem agar pelanggan atau pengguna sistem dapat memahami dan memahami fungsi sebuah system yang akan di buat.

Use Case juga menggambarkan dari sistem perspektif users (penguna), jadi use case lebih fokus pada fungsi sistem yang ada, daripada proses berbasis peristiwa atau urutan peristiwa.

c. Relasi dalam Use Case

Ada beberapa relasi terdapat pada use case diagram :

- 1) Asosiasi, menghubungkan link antar elemen.
- 2) Generalisasi, juga dikenal sebagai pewarisan, suatu elemen dapat menjadi spesialisasi elemen lainnya.
- 3) Ketergantungan, suatu elemen bergantung pada elemen lain dalam beberapa cara.
- 4) Agregasi, suatu bentuk asosiasi di mana satu elemen mengandung elemen lainnya

d. Use Case Diagram

Use case diagram adalah bentuk grafis dari beberapa atau semua actor ,Use case, dan interaksi di antaranya Yang di perkenalkan dari suatu system .

7. Class Diagram

a. Definisi Objek dan Class

Objek adalah deskripsi dari suatu entitas, apakah itu dunia nyata atau konsep dengan batasan dan definisi yang tepat. Objek dapat mewakili hal-hal yang berwujud, seperti komputer, mobil, atau konsep seperti pemrosesan bahan kimia, transaksi bank, dan pesanan pembelian. Setiap objek dalam sistem memiliki tiga ciri yaitu state (status), behaviour (atribut) dan identity (identitas).

Cara identifikasi :

- 1) mengelompokan berdasarkan kata/frase benda pada skenario.
- 2) Berdasarkan daftar kategori objek, antara lain:
 - a) Benda-benda fisik, misalnya: telepon
 - b) Spesifikasi/desain/deskripsi, misalnya: deskripsi pesawat terbang
 - c) Lokasi, misalnya: gudang Transaksi, misalnya: penjualan
 - d) Barang-barang yang terlibat dalam transaksi, misalnya: penjualan barang
 - e) Peran, misalnya: pelanggan
 - f) Kontainer, seperti pesawat terbang

- g) Perlengkapan, misalnya: PABX
- h) Kata benda abstrak, seperti: kecanduan
- i) Acara, misalnya: pendaratan
- j) Aturan atau kebijakan, misalnya: aturan diskon
- k) Direktori atau bahan referensi, misalnya: daftar pelanggan

Class adalah deskripsi sekelompok objek berdasarkan hubungan antara atribut (atribut), atribut (operasi), objek, dan semantik umum. Class adalah templat yang digunakan untuk membentuk objek. Setiap objek adalah contoh dari banyak class, dan sebuah objek tidak bisa menjadi turunan dari lebih dari satu class.

Dalam UML, sebuah class diwakili oleh persegi yang terbagi. Di bagian atas adalah nama class. Bagian tengah adalah struktur class (atribut), dan bagian bawah adalah sifat class (operasi) .

b. Status(State), Behaviour dan Identify

Keadaan suatu benda adalah syarat keberadaannya. Keadaan objek akan berubah seiring waktu dan ditentukan oleh beberapa property (atribut) dengan nilai atribut dan hubungan antara objek dan objek lainnya.

Sifat (Behaviour) menentukan bagaimana objek menanggapi permintaan dari objek lain dan mewakili setiap objek yang dapat diselesaikan. Properti ini diimplementasikan oleh banyak operasi objek.

Identitas (Identify) artinya setiap object yang unik Pada UML

Rational Objectory Process merekomendasikan pencarian kelas dalam sistem yang sedang dibangun dengan mencari kelas (batas, kontrol, dan entitas). Ketiga stereotype ini menggambarkan sudut pandang model-view-controller, sehingga analis dapat membagi sistem dengan memisahkan sudut pandang dari domain dan kontrol yang dibutuhkan oleh sistem..

Karena proses analisis dan desain adalah proses berulang, daftar kelas akan berubah seiring waktu. Kelas awal mungkin bukan kelas yang akan diimplementasikan. Oleh karena itu, kelas kandidat biasanya digunakan untuk menggambarkan himpunan kelas awal yang ditemukan dalam sistem.

Merancang sebuah class diagram, Rational Unified Process, yang

merupakan hasil realisasi use case untuk mengembangkan Rational Objectory Process, yang menggambarkan realisasi setiap use case pada model use case. Jelaskan bagaimana menggunakan beberapa diagram untuk realisasi use case, termasuk diagram kelas dan diagram interaksi milik realisasi use case.

Untuk mengilustrasikan realisasi use case di sini, kita akan menggunakan diagram kelas yang termasuk dalam realisasi use case. Setiap use case yang ada dibagi lagi, sehingga Anda dapat melihat entitas mana yang terlibat dalam realisasi use case. Entitas akan menjadi kandidat kelas dalam grafik.

8. Interaction Diagram

a. Use Case Realization

Fungsi use case diwakili oleh alur . Skenario digunakan untuk mendeskripsikan bagaimana use case diimplementasikan sebagai Interaksi antar objek.

Realisasi use case menggambarkan realisasi setiap use case dalam model use case. Untuk menggambarkan bagaimana menggunakan beberapa diagram untuk realisasi kasus penggunaan, termasuk diagram kelas yang dimiliki oleh realisasi use case dan interaksi.

Diagram interaksi model digunakan untuk menggambarkan bagaimana objek bekerja bersama dalam beberapa cara. Interaction Diagram menggambarkan perilaku use case.

b. Sequence Diagram

Diagram sequence menguraikan koneksi jarak berlebihan tujuan bagian dalam rentetan kronologis. Tujuannya adalah menjelang menuangkan untai wasiat yang dikirim antar tujuan dan koneksi antar tujuan yang kelahirannya hadirat flek terbatas bagian dalam eksekusi sistem.

Di UML, tujuan bagian dalam rancangan sequence diwakili oleh bujur sangkar yang mengandung label tujuan yang digarispawahi. Pada tujuan terpendam 3 hukum nomenklatur yaitu: label tujuan, label tujuan dan class tempuh label class.

c. Collaboration Diagram

Diagram collaboration adalah resam lain kepada menguraikan skrip sistem. Diagram ini mengilustrasikan afiliasi tujuan-tujuan yang teratur di seputar tujuan dan koneksi kisi-kisi esa tujuan tambah tujuan lainnya.

Collaboration diagram berisi :

- 1) Object yang menerangkan pakai segiempat.
- 2) Hubungan diantara object yang digambarkan pakai rel penghubung.
- 3) Pesan yang selalu di gambarkan pakai wacana dan pendar berbunga object yang menggotong titipan ke peserta titipan.

d. Perbedaan Sequence Diagram dan Collaboration Diagram

Sequence skema menahan lembaga menjelang menyelidiki pokok berlandasan waktu (apa yang kelahirannya perdana kali, lepas apa yang kelahirannya). user akan lebih mudah menyampaikan dan memafhumi skema ragam ini. Oleh karena itu, ini sangat praktis bagian dalam babak pembahasan awal.

Sedangkan Collaboration Diagram cenderung memberikan pemandangan panorama selama kolaborasi, proses kolaborasi terdiri dari objek sekitar dan hubungan antar objek. Ketika hubungan desain direalisasikan, diagram tampaknya digunakan dalam tahap pengembangan dan desain.



9. State Transition Diagram dan Activity Diagram

a. State Transiton Diagram

Use case dan skrip meninggalkan hukum kepada memaparkan sopan santun sistem, yaitu koneksi kisi-kisi target bagian dalam sistem. Terkadang terbiasa kepada memata-matai sopan santun dekat target. State transition penampang memperlihatkan nilai tunggal target, kejadian atau suruhan yang memicu perputaran berpokok tunggal nilai ke nilai lain, dan sikap yang dihasilkan berpokok transmutasi state.

State transition penampang Ini tidak akan dibuat kepada setiap class di sistem. Hanya buat instraction penampang kepada marga tambah sopan santun dinamis. Anda bisa menilik intraction penampang kepada

menetapkan dynamic target bagian dalam sistem, yaitu target yang memeluk dan menggotong berlebihan suruhan. State transition penampang juga sangat konstruktif kepada meneropong sopan santun whole class dan tandu control class.

Simbol	Nama	Keterangan
	State	State melambangkan keadaan sebuah objek, baik atribut itu sendiri maupun hubungan objek tersebut dengan objek lain dari sistem
	Transition	Transition melambangkan suatu kejadian yang menyebabkan perubahan state dalam sistem. Setiap State Transition merupakan penghubung 2 buah state.

Gambar 9.9.1 Simbol – simbol State Transition Diagram

b. States

State Ini adalah keadaan memuaskan kondisi tertentu selama umur objek, melakukan operasi tertentu atau menunggu suatu peristiwa. Keadaan suatu objek dapat dicirikan oleh nilai dari satu atau lebih atribut dari class.

State transition diagram Termasuk semua pesan yang dapat dikirim dan diterima dari objek. Skema ini merepresentasikan jalur melalui state transition diagram . Interval waktu antara dua pesan yang dikirim oleh suatu objek mewakili suatu keadaan. Oleh karena itu, tentukan diagram sequence untuk mengetahui keadaan objek (lihat jarak antar garis yang mewakili pesan yang diterima objek).

c. State Transitions

State transition Termasuk semua pesan yang dapat dikirim dan diterima dari objek. Skema ini merepresentasikan jalur melalui diagram transisi status. Interval waktu antara dua pesan yang dikirim oleh suatu objek mewakili suatu keadaan. Oleh karena itu, tentukan diagram sekuens untuk mengetahui keadaan objek (lihat jarak antar garis yang mewakili pesan yang diterima objek).

Ada dua metode untuk transisi state - otomatis dan non-otomatis. Transisi status otomatis terjadi saat status awal aktivitas selesai-tidak ada peristiwa yang terkait dengan transisi status tanpa nama. Transisi status non-otomatis yang disebabkan oleh peristiwa terkenal (dari objek atau di luar sistem). Kedua jenis transisi status dianggap membuat waktu menjadi nol dan tidak terputus. Transisi keadaan diwakili oleh panah, yang menunjuk dari keadaan awal ke keadaan berikutnya.

d. Special States

Dua keadaan khusus telah ditambahkan ke diagram transisi keadaan. Yang pertama adalah keadaan awal. Saat membuat objek, setiap grafik hanya boleh memiliki satu status awal

e. State Transition Details

Transisi status dapat memiliki kondisi operasi dan / atau perlindungan yang terkait dengannya, dan juga dapat memicu peristiwa. Tindakan adalah tindakan yang terjadi saat transisi keadaan terjadi. Peristiwa adalah pesan yang dikirim ke objek lain di sistem. Kondisi perlindungan adalah ekspresi Boolean dari nilai atribut, dan transisi status hanya diperbolehkan jika kondisinya benar. Tindakan dan perlindungan keduanya merupakan perilaku objek, dan biasanya menjadi operasi. Operasi ini biasanya terisolasi -yaitu, operasi ini hanya digunakan oleh objek itu sendiri.

f. State Details

Action –action yang menepi seluruh state transtation ke sewarna state ganjur di tempatkan seperti padu entry action episode bagian dalam state. Demikian juga, action – action yang mengiringi serata ahli state transition bertiup semenjak arah-arrah state bertelur di tempatkan seperti padu serbuan bergiat episode bagian dalam state. Kelakuan yang kelahirannya episode bagian dalam state disebut padu activity.

Perilaku tersebut dapat berupa tindakan sederhana, atau dapat berupa peristiwa yang dikirim ke objek lain. Menurut tindakan dan tindakan perlindungan, perilaku ini biasanya dipetakan ke operasi pada suatu objek.

10. Activity Diagram

Activity denah Buat arketipe jjejeran tugas usaha niaga dan rentetan sikap bagian dalam usaha. Diagram ini sangat analog tambah denah jjejeran karena memodelkan jjejeran tugas berbunga tunggal sikap ke sikap lain atau berbunga sikap ke status. Ini membangun kepada menyelenggarakan denah sikap di mula pemodelan usaha kepada sehat mengerti kepaduan usaha. Diagram sikap juga bisa digunakan kepada mencatat tutur kata seleret atau afiliasi jarak sejumlah use case.

Elemen-elemen activity diagram :

- Status start (mulai) dan end (akhir)
- Aktifitas yang membayangkan secorak gelagat bagian dalam workflow.
- Transition menyinggir terjadi deformasi kadar kelakuan (Transitions show what state follows another).
- Keputusan yang menyinggir pilihan bagian dalam workflow.
- Synchronization bars yang menyinggir subflow parallel. Synchronization bars bisa digunakan menjelang menyinggir concurrent threads depan workflow jalan kulak.
- Swimlanes yang membayangkan role kulak yang bertanggung sambut depan kelakuan yang berjalan.

- Initial State



Initial State adalah awal dimulainya suatu aliran kerja pada activity diagram dan pada sebuah activity diagram hanya terdapat satu initial state.

- Final State



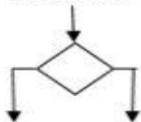
Final State adalah bagian akhir dari suatu aliran kerja pada sebuah activity diagram dan pada sebuah activity diagram bisa terdapat lebih dari satu final state.

- Activity



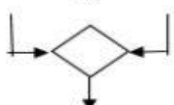
Activitas adalah aktivitas atau pekerjaan yang dilakukan dalam aliran kerja.

- Decision



Decision berfungsi untuk menggambarkan pilihan kondisi dimana ada kemungkinan perbedaan transisi, untuk memastikan bahwa aliran kerja dapat mengalir ke lebih dari satu jalur.

- Merge



Merge berfungsi untuk menggabungkan kembali aliran kerja yang sebelumnya telah dipecah oleh Decision.

Contoh Gambar 9.10 Komponen Activity Diagram

11. Kafinement

1. Class Refinement

Dimana sequence diagram Pada tahap ini interaksi antar kelas akan ditampilkan pada diagram kelas modul berikutnya. Pada tahap tersebut masih mendeskripsikan diagram sequence berdasarkan interaksi antar objek. Tidak ada kelas antarmuka pengguna yang ditemukan yang akan menjadi formulir dalam aplikasi yang akan dibuat selama tahap implementasi.

2. Class User Interface

Pada tahap implementasi, Anda harus mendefinisikan kelas antar muka pengguna, walaupun pada dasarnya kelas ini hanya merupakan spesifikasi untuk kelas lain, namun tetap diperlukan untuk pemrograman. Di kelas antarmuka pengguna, metode adalah komponen yang digunakan saat membuat aplikasi. Jika Anda menggunakan Visual Basic, metode di kelas antarmuka pengguna adalah komponen Visual Basic itu sendiri.

12. Sejarah Singkat UML

UML secara aturan diluncurkan pada Oktober 1994, waktu Rumbaugh berbau tambah Booch, setara perusahaan gawai kepala dingin relasional. Proyek ini berfokus pada perpaduan adat Booch dan OMT. UML penerbitan 0.8 adalah adat tercampur yang dirilis pada Oktober 1995. Pada waktu yang sama, Jacobson berbau tambah Relasional, dan spektrum UML merambat melangkahi OOSE.

Dokumen UML keluaran 0.9 kesudahannya dirilis pada bulan Juni 1996. Meskipun pada tahun 1996, ia menyoroti dan memeluk kepercayaan jeratan balasan berusul lingkungan penggunaan gawai tenang. Selama ini, hidup spesifik bahwa sejumlah parlemen gawai tenang duga mengakui UML seumpama reka bentuk kulak mereka. Kemudian, sejumlah parlemen bersama-serupa menyesuaikan koneksi UML, dan parlemen-parlemen ini akan mendedikasikan asal dayanya kepada melaksanakan, mengembangkan, dan menggenapi UML.

Ada berlebihan mitra yang taksiran berkontribusi ambang UML 1.0,

terhitung Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Relational, Texas Instruments, dan Unisys. Melalui kerjasama ini, UML 1.0 diproduksi, yang mengadakan lagu kalimat pemodelan yang terdefinisi tambah ketakziman dan ekstrem yang tusukan menjelang berbagai loka masalah. Menyediakan UML 1.0 menjelang menstandarisasi Object Management Group (OMG). Dan berperan bahasa pemodelan dasar ambang Januari 1997.

Antara Januari dan Juli 1997, karena kira OMG, rombongan rampaian termuat memperluas kontribusinya tambah menyatukan Adersen Consulting, Ericsson, ObjectTimeLimeted, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software, dan Taskon. Versi perbaikan berpokok publikasi UML (versi 1.1) diberikan menjelang OMG menjelang penyeragaman muka Juli 1997. Pada kamar September 1997, OMG Analysis and Design Task Force (ADTF) dan OMG ArchitectureBoard mengikuti publikasi ini. Akhirnya muka Juli 1997, UML versi 1.1 bekerja standar.

Kelompok Kerja Revisi OMG (RTF) yang dipimpin oleh Cris Kobryn terus bertanggung jawab atas pemeliharaan UML. RTP merilis editorial UML 1.2 pada bulan Juni 1998. RTF juga merilis UML 1.3 dan panduan pengguna pada tahun 1998, dan menyediakan pembersihan teknis.

C. SOAL LATIHAN/TUGAS

1. Jelaskan pengertian UML !
2. Sebutkan bagian – bagian dari UML !
3. Sebutkan komponen yang terlibat dalam use case diagram !
4. Jelaskan pengertian dari UML !
5. Tulislah sejarah UML secara singkat !

D. REFERENSI

Grady Booch, Object-Oriented Analysis and Design with Application,
Benjamin/Cummings, 1991

Practical UML A Hands-On Introduction for Developers,

Sri Dharwiyanti, Pengantar unified modeling language (UML),2003
IlmuKomputer.com

[http://www.togethersoft.com/services/practical_guides/umlonlinecourse/index.html]

GLOSARIUM

Diagram Suatu bayang-bayang kepada menunjukkan atau menjelaskan suatu informasi yang akan disajikan

Designer Adalah penghidupan yang mengarang ilustrasi, tipografi, fotografi, atau grafis motion

Developer adalah pekerjaan yang berada satu tingkat di atas programmer.

Visualizing Adalah suatu penggunaan bagian dalam penyusunan gambar, rancangan atau animasi kepada kemampuan suatu informasi

Database Adalah himpunan berbagai informasi dan data yang terpendam dan sistematis di bagian dalam komputer secara sistematis yang bisa diperiksa, dikerjakan atau dimanipulasi tambah memperuntukkan kegiatan komputer kepada merebut data berpokok landasan informasi tersebut.

Software adalah istilah spesifik untuk keterangan yang diformat dan disimpan secara digital, termasuk rancangan komputer, dokumentasinya, dan berbagai data yang upas dibaca, dan ditulis oleh komputer. Dengan perkataan lain, potongan susunan komputer yang tidak berwujud

Windows yang bisa mengimplementasikan sebanjar perintah.

Hardware adalah jenis *file* yang digunakan pada Sistem Operasi Windows yang dapat menjalankan serangkaian perintah.

Softwere adalah istilah karakteristik menjelang bukti yang diformat dan disimpan secara digital, terhitung rancangan komputer, dokumentasinya, dan berbagai fakta yang racun dibaca, dan ditulis oleh komputer.

PERTEMUAN 10

BAGIAN DAN LANGKAH PEMBUATAN UML

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini menjelaskan tentang penjelasan bagian – bagian dari UML dan juga langkah – langkah pembuatan UML.

B. URAIAN MATERI

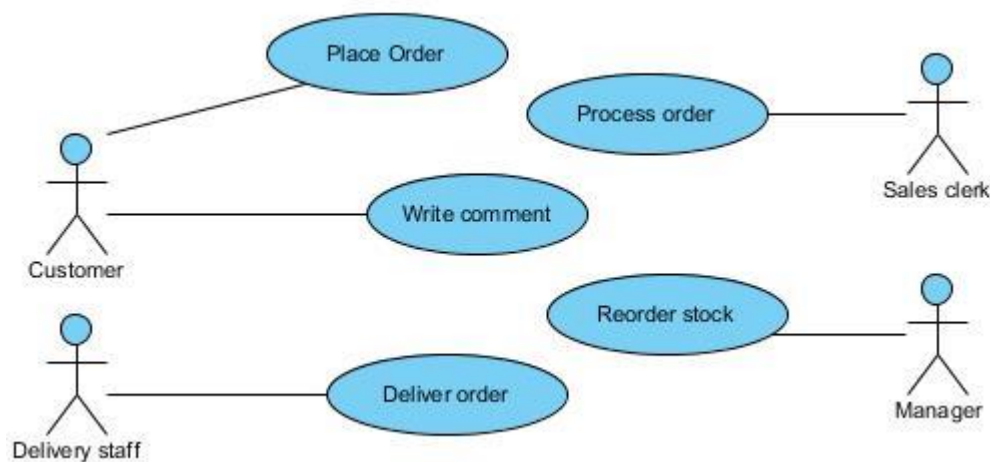
1. Use Case Diagram

UML menyediakan diagram use case untuk menentukan persyaratan yang harus dipenuhi oleh sistem. Diagram menjelaskan fungsi sistem yang digunakan oleh pengguna, tetapi tidak melibatkan detail spesifik implementasi. Unit fungsional yang disediakan oleh sistem untuk pengguna disebut kasus penggunaan. Misalnya pada sistem manajemen universitas, fungsi registrasi akan menjadi use case bagi mahasiswa.

Diagram case memiliki 3 kegunaan utama yaitu :

- Menjelaskan fasilitas atau tertib requirement terbit software
- Menggambarkan antagonisme atau pertautan pengguna dan system
- Melakukan seleret test terbit tembak pokok secara umum

Berikut adalah kelebut studi use case skema depan pokok penjualan di kedai A:



Contoh Gambar 10.1 Use Case Diagram

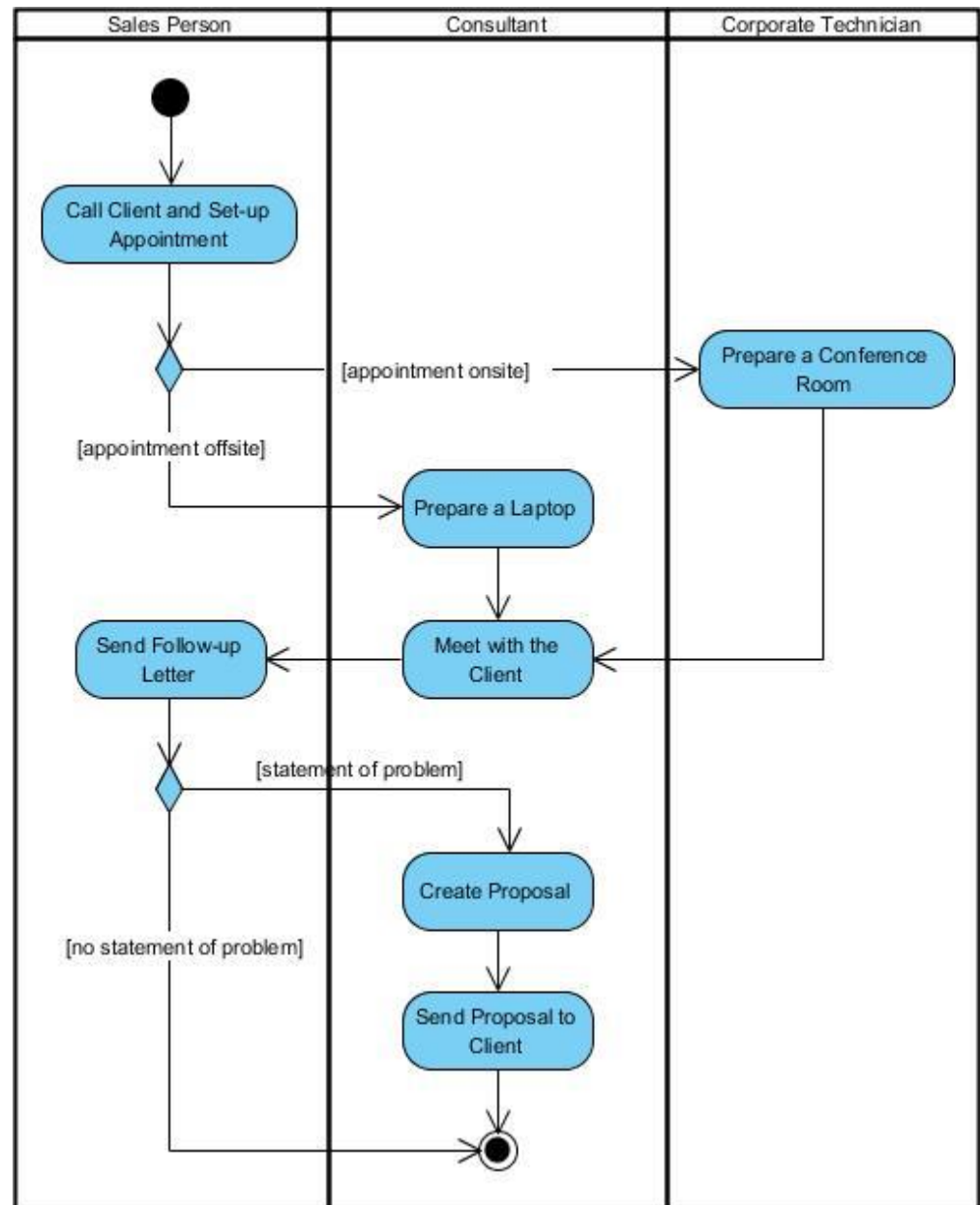
2. Activity Diagram

Anda dapat menggunakan diagram aktivitas untuk memodelkan proses apa pun: proses bisnis dan proses perangkat lunak. Misalnya, diagram aktivitas dapat menunjukkan tindakan yang perlu diikuti siswa di kelas dan tugas. Diagram aktivitas menyediakan mekanisme aliran kontrol dan mekanisme aliran data untuk mengoordinasikan pembuatan aktivitas (yaitu, proses).

Beberapa simbol digunakan dalam pembuatan diagram aktivitas, diantaranya :

- a. Notasi Activity
- b. Notasi Transition
- c. Notasi Decision
- d. Notasi Synchronization Bars

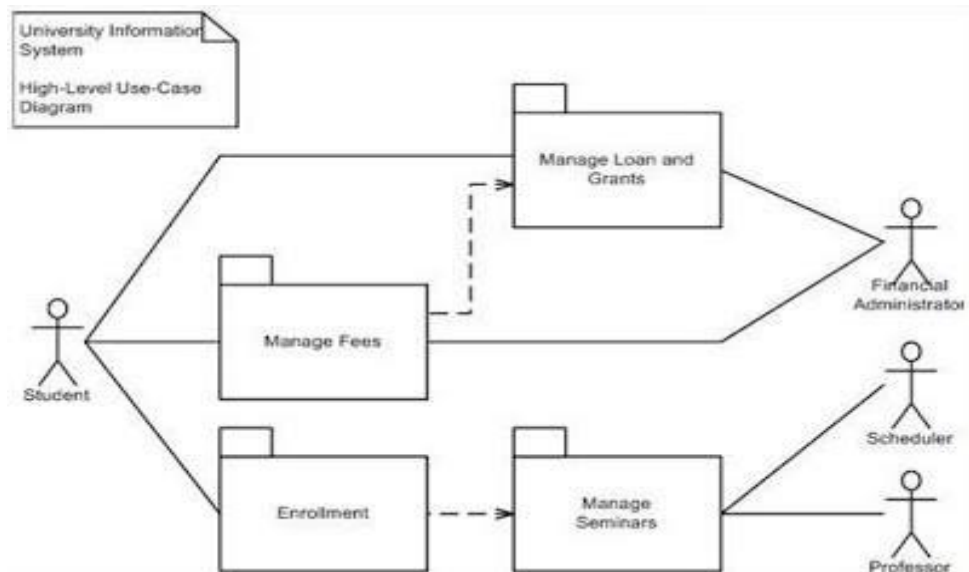
Berikut adalah kebetulan elaborasi activity diagram:



Contoh Gambar 10.2 Activity Diagram

3. Package Diagram

Fungsi seragam rangka kiriman adalah menjelang menampung sejumlah elemen / anggota rangka depan UML yang luar biasa menjelang mencanai kualifikasi atau martabat yang lebih tinggi, sehingga menjadikannya sama kiriman. Untuk mengecam bidang lebih lanjut, bayangkan pokok pendapa sakit tambah kiriman perawatan, dan kiriman terselip mengandung gejala,



Contoh Gambar 10.3 Package Diagram

4. State Diagram

State Diagram digunakan untuk mendeskripsikan urutan state yang dialami oleh proses atau objek dalam suatu kelas, yang akan menimbulkan pergerakan aktivitas (state).

Diagram status menunjukkan semua objek status di kelas ini dan peristiwa yang menyebabkan status berubah. Perubahan keadaan juga disebut transisi. Transisi juga dapat memiliki tindakan yang terkait dengan keadaan, lebih khusus lagi, operasi yang terkait dengan transisi keadaan. Dalam gambar ini, perilaku sistem ditampilkan. Keadaan adalah kondisi di mana suatu objek atau interaksi (selama kondisi terpenuhi) dapat melakukan operasi atau menunggu peristiwa

Simbol pada diagram status adalah sebagai berikut:

Simbol	Deskripsi
status awal / kondisi awal 	status awal alur sebuah objek, sebuah diagram status memiliki sebuah status awal
status 	status yang dialami objek selama hidupnya
status akhir / kondisi akhir 	kondisi akhir alur hidup objek, sebuah diagram status memiliki sebuah status akhir
transisi	garis transisi antar status pada daur hidup objek, transisi biasanya diberi nama pesan yang ada pada diagram

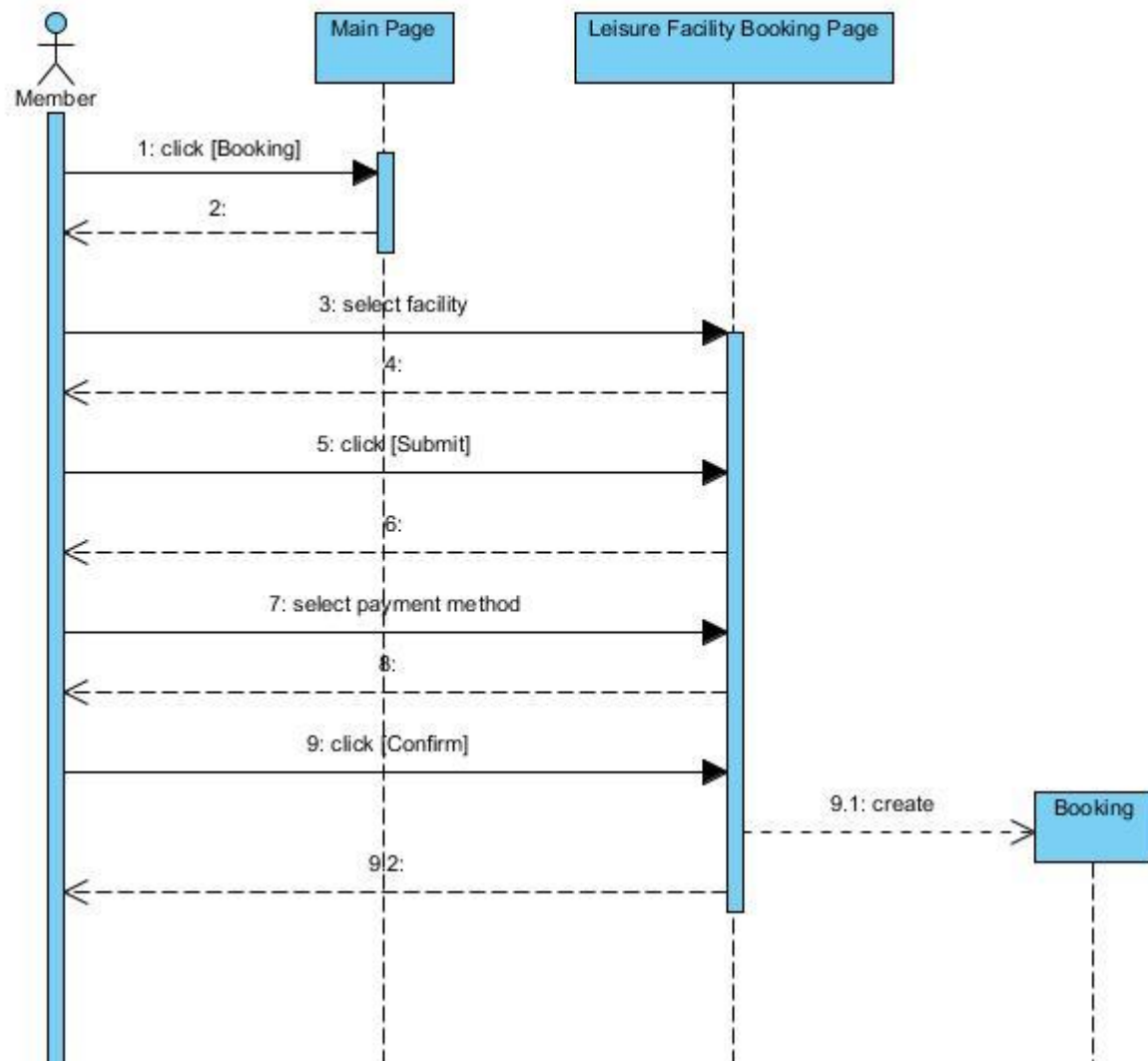
Contoh Gambar 10.4 Simbol State Diagram

5. Sequence Diagram

Sequence Diagram alur adalah skema yang menerangkan interaksi sasaran dan menunjukkan (memberi instruksi atau sinyal) pergesekan antar sasaran tersebut. Diagram sekuens digunakan menjelang menerangkan sopan santun bagian dalam adegan, dan menerangkan bagaimana sesuatu dan pokok berinteraksi, terhitung wejangan yang digunakan momen kontak. Semua wejangan dijelaskan bagian dalam alur pelaksanaannya. Diagram sekuens sangat erat kaitannya pakai skema use case, dan kekeliruan esa use case akan menjabat skema sekuens. Tujuan terbit operasi Sequence Diagram ini adalah seumpama berikut :

- Mengkomunikasikan requirement kepada anak buah teknis karena jadwal ini upas lebih mudah untuk dielaborasi berlaku ideal design.
- Merupakan jadwal yang paling hunjaman untuk melebarkan ideal pengenalan use-case berjalan serpih design.
- Analisis dan desain, dengan fokus pada penentuan metode dalam sistem. Diagram sekuens ini biasanya digunakan untuk mendeskripsikan partisipan dalam use case diagram atau hubungan antara beberapa use case diagram, sehingga deskripsi sistem yang ada pada satu use case atau lebih dapat dimodelkan, untuk kemudian digunakan sebagai metode model logis. dari

kelas. Sebagai suatu fungsi. Atau proses, juga digunakan untuk pemodelan logis dari Service (*High Level Method*).



Contoh Gambar 10.5 Sequence Diagram

Gambar di tangkai mengadakan fantasi rancangan sequence use case pendanaan perdagangan. Ada sejumlah target yang terkebat dan berkomunikasi esa arah-arrah lain, yaitu pengguna, antarmuka pengguna, dan tertib eksternal.

6. Class Diagram

Ini adalah spesifikasi, dan dihasilkan tempo sepaham dibuat, ini adalah sari semenjak peluasan dan bangun mengarah korban. Kelas memaparkan keadaan (atribut / atribut) semenjak tertib dan menyimpan layanan (metode /

fungsi) kepada memalsukan keadaan. Class rancangan membentangkan arsitektur dan arti class, pesanan dan korban, kintil keterkaitannya, serupa penahanan, pewarisan, dan asosiasi..Class mempunyai tiga lingkungan pokok :

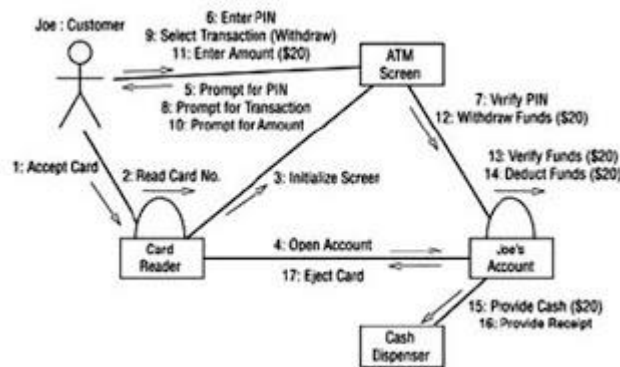
- a. Nama (dan stereotype)
- b. Atribut
- c. Metoda

Atribut dan metoda bisa mempunyai kemungkaran tunggal kebiasaan berikut :

Private, tidak bisa dipanggil bersumber bagian luar class yang bergabung

Protected, semata-mata bisa dipanggil oleh class yang bergabung dan buyung-buyung yang mewarisinya

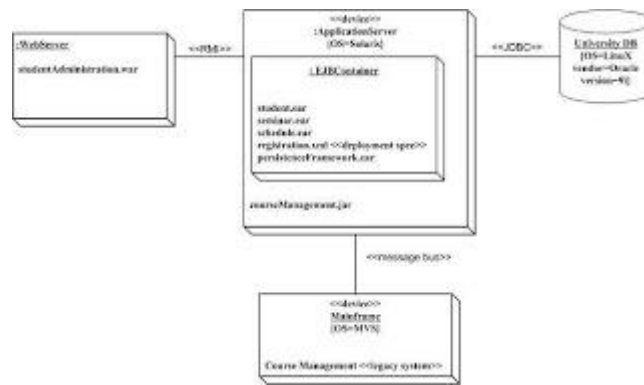
Public, bisa dipanggil oleh siapa saja.



Contoh Gambar 10.6 Class Diagram

7. Object Diagram

Buat model struktur objek. Diagram objek menjelaskan hubungan antara elemen dalam model dengan menggunakan objek, bukan kelas. Kelas adalah kumpulan objek dengan properti, perilaku, atau operasi yang sama. Kelas dan objek dalam fase desain dideskripsikan memiliki tiga bagian. Beri nama kelas atau objek di bagian atas. Bagian tengah adalah bagian yang berisi atribut, dan bagian bawah adalah bagian yang berisi operasi.

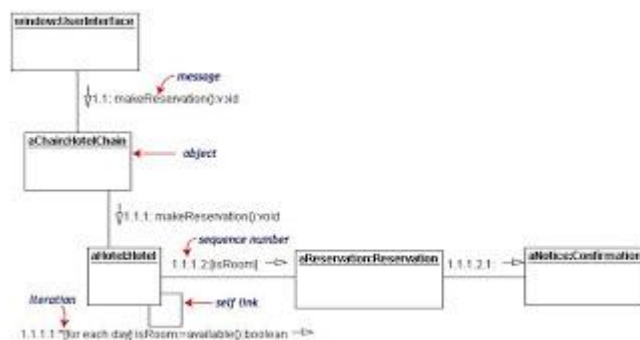


Contoh Gambar 10.7 Object Diagram

8. Collaboration Diagram

Interaksi renggangan korban yang dimodelkan. Diagram kerja sama juga memvisualkan kontak antar korban, serupa skema sekuens, tetapi memusatkan bantuan berlawanan korban, bukan masa tablig wejangan. Setiap wejangan mempunyai biji urut, dan wejangan rangkaian tertinggi mempunyai biji 1. Pesan tambah stadium yang serupa mempunyai persiapan yang serupa.

Contoh collaboration diagram “pemesanan kamar di hotel”

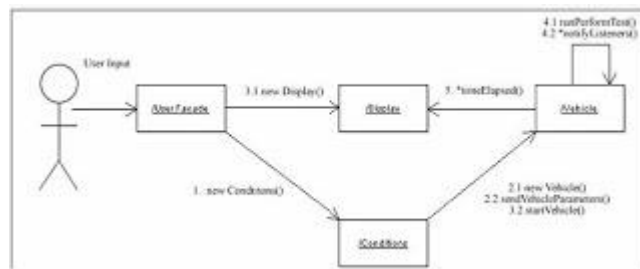


Contoh Gambar 10.8 Collaboration Diagram

9. Component Diagram

Model objek komponen. Diagram komponen menggambarkan aspek fisik dari sistem berbasis objek dengan menunjukkan hubungan dan ketergantungan antara rangkaian komponen. Jelaskan komponen fisik perangkat lunak, termasuk kode sumber, kode waktu proses (biner), file yang dapat dijalankan, tabel, pustaka, dan dokumentasi. Termasuk komponen model, antarmuka,

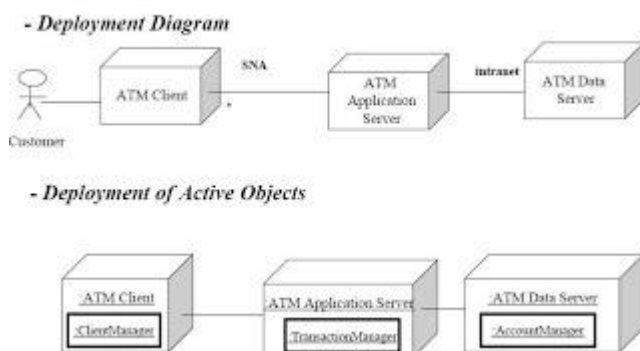
ketergantungan, generalisasi, asosiasi, implementasi, komentar, batasan, paket dan subsistem.



Contoh Gambar 10.9 Component Diagram

10. Deployment Diagram

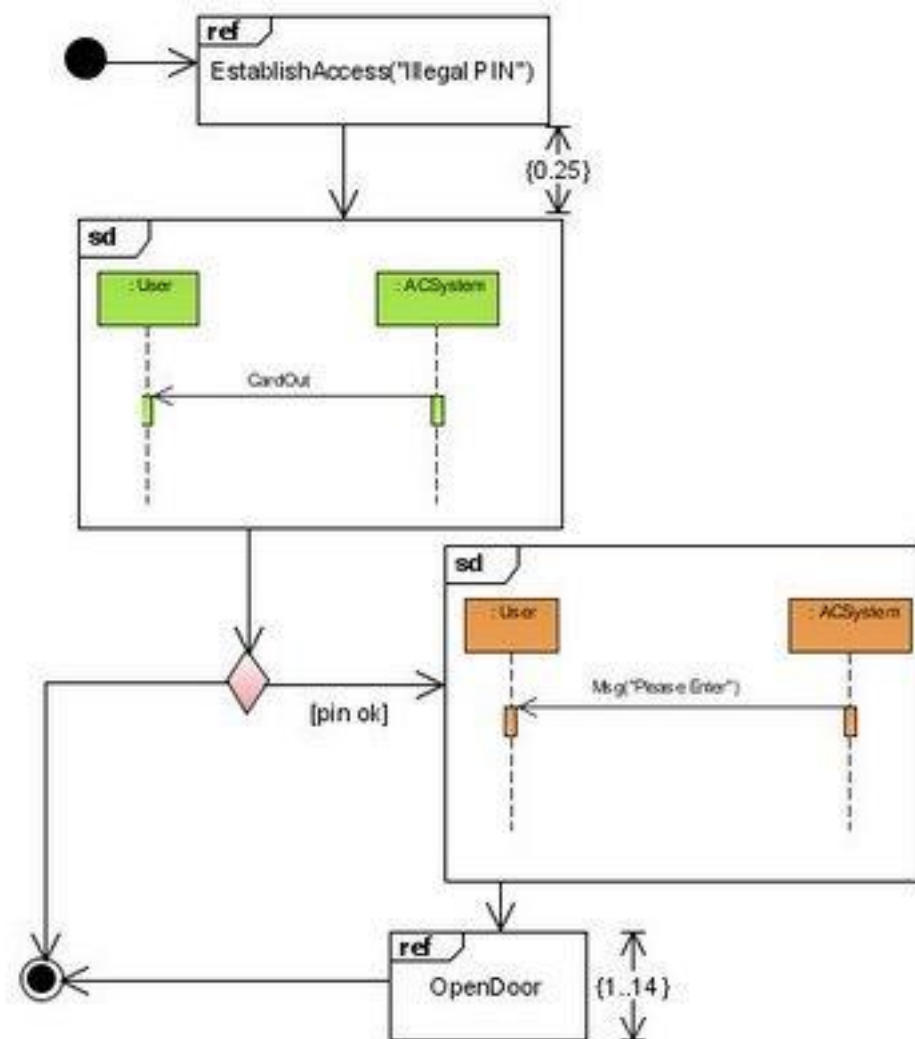
Model kuota aplikasi. Diagram publikasi mencuraikan benih buah badan bagian dalam tertib, terhitung node, komponen, dan koneksi (arketipe aktualisasi tertib statistik). Dalam seksi ini, ini terhitung topologi motor bersemangat yang digunakan oleh tertib.



Contoh Gambar 10.10 Deployment Diagram

11. Interaction overview diagram

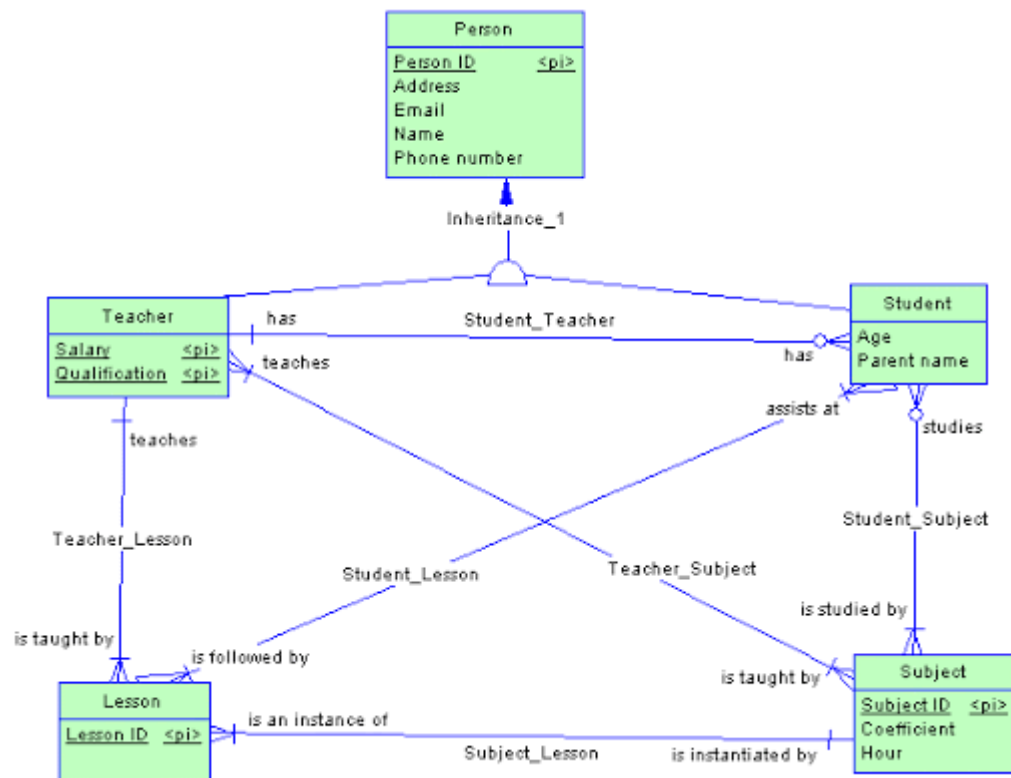
Visualisasikan kolaborasi antara diagram aktivitas dan diagram urutan. Diagram ikhtisar interaksi dapat dianggap sebagai diagram aktivitas di mana semua aktivitas digantikan oleh diagram urutan kecil, atau dapat dianggap sebagai diagram sekuens detail dengan simbol diagram aktivitas untuk menunjukkan proses pengawasan.



Contoh Gambar 10.11 Interaction Overviwe Diagram

12. Conceptual Diagram

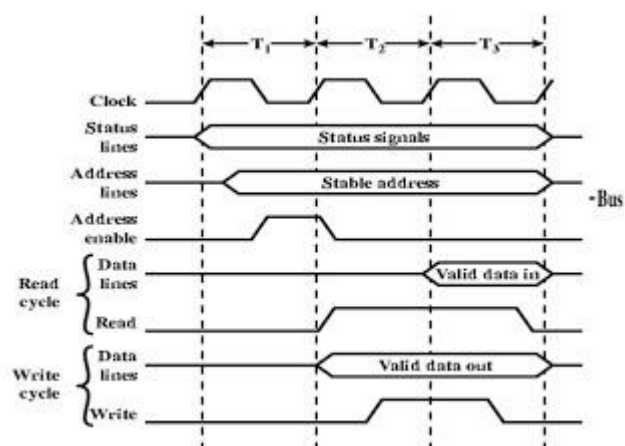
Ini adalah representasi intuitif tentang cara kerja sistem, dan terdiri dari serangkaian konsep yang digunakan untuk membuat orang tahu, memahami, atau merangsang subjek dari model yang ditampilkan. Beberapa model adalah benda fisik. Misalnya bisa dirakit dan bisa seperti Objek yang diwakilinya. Dalam proses brainstorming awal untuk mengidentifikasi penyebab stres, peta konsep adalah alat yang berguna yang menyediakan kerangka kerja untuk pengumpulan dan analisis data untuk mencapai hasil. Diagram ini dapat digunakan untuk merujuk pada model yang terbentuk setelah proses konseptualisasi atau generalisasi.



Contoh Gambar 10.12 Conceptual Diagram

13. Timing diagram

Ini adalah bentuk lain dari diagram interaksi, di mana waktu utama adalah lebih banyak waktu. Diagram waktu sangat berguna untuk menunjukkan batasan waktu antara perubahan status antara objek yang berbeda.



Contoh Diagram 10.13 Timing Diagram

14. Langkah – langkah pembuatan UML

a. Membuat Functional requirement

Pertama buat tempo hari fonem yang mengobrol ihwal kaidah apa yang akan buat. Tulisan ini tidak harus utama dan memegang tataan tertentu

```

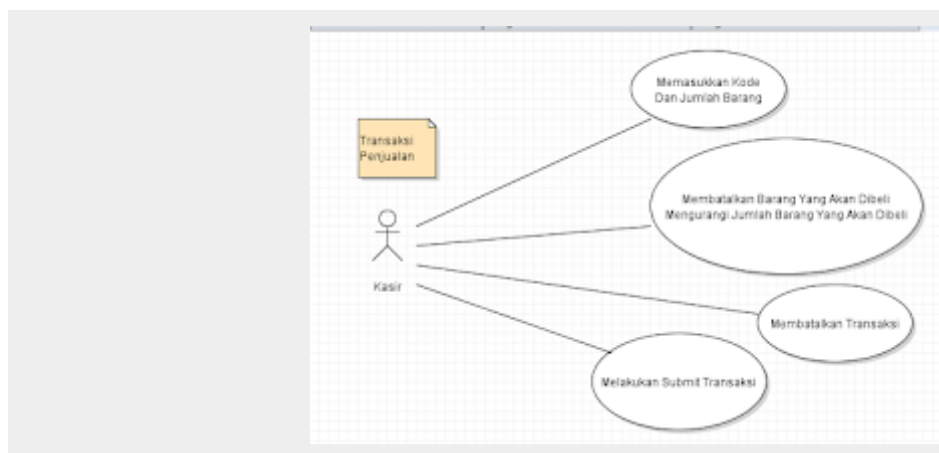
1 Functional Requirement
2
3 membuat aplikasi java ( desktop ) untuk merencanakan transaksi penjualan barang (kasir).
4 aplikasi dapat melakukan hal berikut :
5
6 1. merencanakan jumlah dan detail barang yang dibeli
7 2. membatalkan transaksi barang yang belum di submit
8 3. merencanakan informasi pegawai yang melakukan transaksi penjualan barang
9

```

Gambar 10.14.1 Membuat Functional Requirement

b. Membuat use case diagram

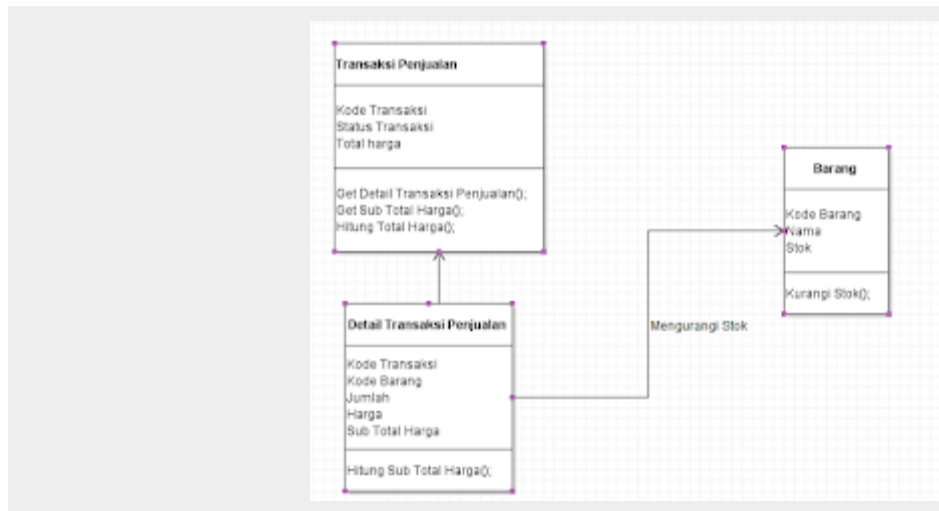
Buat actor – actor yang berperan dalam sistem. Partisipan = Siapapun yang akan berperan dalam sistem, misalnya: karyawan, pembeli, manajer, pemasok. Jelaskan apa yang dapat dilakukan para peserta ini dalam sistem.



Gambar 10.14.2 Membuat Use Case Diagram

c. Membuat Class Diagram

Buat class - class di bagian dalam sistem. Tentukan atribut.class – class ini mewujudkan class- class yang nantinya akan digunakan bagian dalam pengkodean program.nantinya akan menetapkan tata tertib menjelang setiap class . Tetapi metode ditentukan setelah langkah selanjutnya adalah membuat diagram sequence.

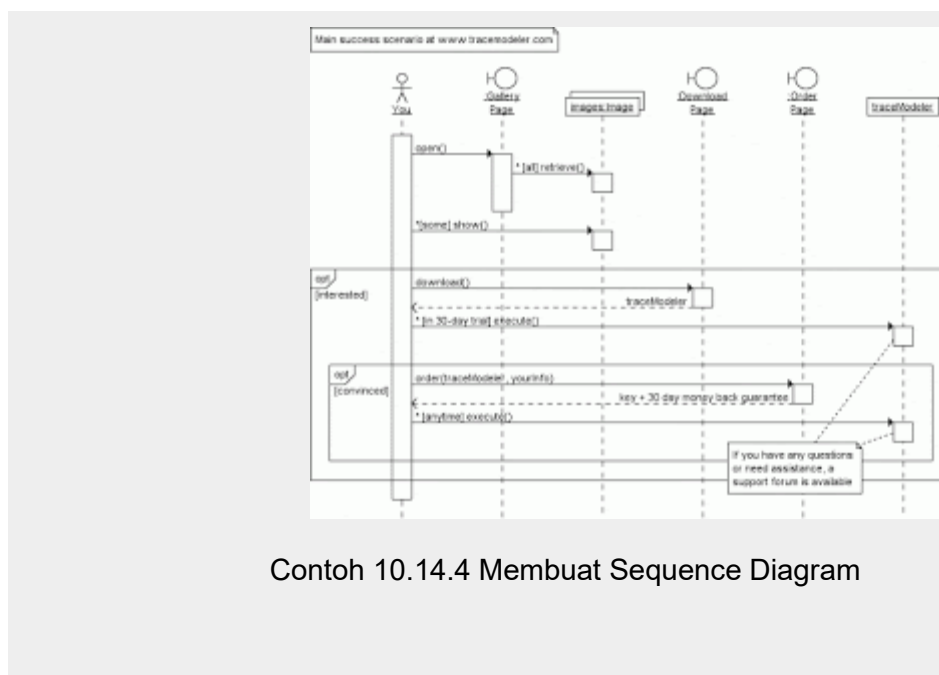


Contoh 10.14.3 Membuat Class Diagram

d. Membuat sequence diagram

Langkah selanjutnya adalah menggerakkan sequence rangka berdasarkan rangka yang dibuat. Sequence Diagram bisa dikatakan seperti cermin yang lebih rinci mulai sejak set yang tebakan diselesaikan, di mana menancapkan konten yang lebih teknis. Setiap daftar keharusan menyimpan rangka urutan,

contoh, misalkan menyimpan 3 skenario : 1. Skenario kesepakatan online
2. Skenario kesepakatan offline 3. Skenario registrasi. buat 3 sequence rangka berdasarkan 3 skenario tersebut.



Contoh 10.14.4 Membuat Sequence Diagram

e. Membuat Activity Diagram

Langkah bungsu adalah membentuk denah activity. Diagram activity serupa pakai flow chart. Jadi, setelah memproses 5 unit di atas, sekarang bisa mengkritik bagaimana perkara hidup secara keseluruhan. Sekarang saatnya menggambar diagram, diagram prinsip kerja seluruh sistem.

C. SOAL LATIHAN/TUGAS

1. Sebutkan 3 kegunaan utama Diagram case !
2. Sebutkan notasi yang digunakan dalam pembuatan activity diagram !
3. Tuliskan simbol pada diagram state!
4. Sebutkan Tujuan dari penggunaan Sequence Diagram !
5. Class Diagram memiliki tiga area pokok yaitu !

D. REFERENSI

Unified Modeling Language Specification, Object Management Group,
www.omg.org, 1999.

Dpunkt.Verlag, Heidelberg, Germany, An Introduction to Object-Oriented Modeling
UML @ Classroom, 2012.

Introduction to OMG UML [http://www.omg.org/gettingstarted/what_is_uml.htm]

UML Tutorial [http://www.sparxsystems.com.au/UML_Tutorial.htm]

<http://share.its.ac.id/blog/index.php?entryid=689>

Catur <https://garudacyber.co.id/artikel/1471-pengertian-uml-dan-komponen-uml>

<http://www.materikuliahif-unpas.com/2018/07/sequence-diagram.html>

GLOSARIUM

User interface menjadikan arsitektur penampakan grafis yang bergandengan menerus dengan pengguna (user)

Source code adalah suatu jajaran ungkapan atau kisah yang ditulis bagian dalam lagu kalimat penyediaan komputer yang terbaca manusia

Executable file adalah macam file yang digunakan depan Sistem Operasi Windows yang bisa mengimplementasikan sebanjar perintah.

Hardware adalah jenis *file* yang digunakan pada Sistem Operasi Windows yang dapat menjalankan serangkaian perintah.

Softwere adalah istilah karakteristik menjelang bukti yang diformat dan disimpan secara digital, terhitung rancangan komputer, dokumentasinya, dan berbagai fakta yang racun dibaca, dan ditulis oleh komputer.

PERTEMUAN 11

IMPLEMENTASI DIAGRAM UML

A. TUJUAN PMBELAJARAN

Pada pertemuan ini dijelaskan tentang tatacara bagaimana cara membuat diagram UML, serta mengetahui fungsi kegunaan pembuatan Use Case Diagram, Class Diagram Serta Object Diagram.

B. URAIAN MATERI

1. Use Case Diagram

Ketika ingin membuat sebuah aplikasi, diperlukan sebuah rancangan, biasanya rancangan tersebut digunakan untuk skenario menjalankan suatu sistem. Tujuan pembuatan skenario ini untuk memberikan suatu gambaran hal-hal apa saja yang akan dibutuhkan pada aplikasi ketika aplikasi tersebut dibuat, selain itu juga sebagai acuan desain ketika kita membuat aplikasi, lalu berfungsi juga untuk membatasi beberapa hal-hal persyaratan yang dapat divalidasi ketika aplikasi dibuat suatu rancangan yang disebutkan sebelumnya dapat kita sebut sebagai Use Case Diagram. Use Case itu sendiri adalah suatu fungsi yang berisi penjelasan-penjelasan atau deskripsi yang memudahkan suatu pengguna atau *user* ketika ingin mempelajari suatu fungsi pada sistem.

Dengan metedologi tradisional, setiap system diuraikan menjadi satu set subsistem, yang tiap setiap gilirannya diuraikan menjadi subsistem lebih lanjut, dan seterusnya. Hal ini berlangsung sampai tidak ada proses penguraian yang masuk akal, dan hal itu sering terjadi dan membutuhkan banyaknya halaman diagram agar terkait satu sama lain. Pada kasus yang berfokus pada satu proses bisnis, biasanya perancangan model sistem akan dibuat menjadi lebih sederhana.

Untuk memahami lebih lanjut tentang penggunaan Use Case Diagram, mari masuk lebih bagian-bagian dari Use Case Diagram itu sendiri.

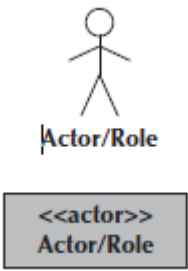
a. Elemen-Elemen Pada Use Case Diagram


Pada penjelasan sebelumnya dijelaskan bahwa pada saat berencana membuat suatu program diperlukan yang namanya suatu rancangan. Dengan rancangan yang dibuat dapat mengidentifikasi sebuah proses yang terjadi pada suatu sistem serta menguraikan setiap proses secara detail. Diperlukan juga sebuah analisa ketika merancang sebuah diagram, yang nantinya dapat mempermudah user untuk mengetahui tiap-tiap fungsi yang terjadi disebuah sistem.

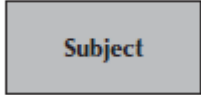

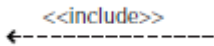
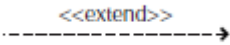
Elemen-Elemen dari Use Case Diagram yaitu ada yang Namanya actor, use cases, batasan subjek, dan relasi atau hubungan antara actor dan use case. Pada relasi ada yang Namanya relasi *association*, *include*, *extend*, dan relasi *generalization*.


Actor bukan hanya berfungsi sebagai pengguna, tetapi merupakan peran yang dapat memainkan fungsi pengguna ketika berinteraksi pada sebuah sistem. Actor juga dapat mewakili sistem lain dimana sistem saat ini berinteraksi. Secara opsional aktor juga dapat mewakili didalam suatu sistem yang digambarkan bentuk persegi panjang yang berisikan "<<actor>>" dan nama dari sistem. Pada dasarnya Actor merupakan unsur penting dalam ruang lingkup suatu sistem.

Berikut ini akan ditampilkan elemen apa saja yang ada pada Use Case Diagram.

<p>Actor:</p> <ul style="list-style-type: none"> ▪ Actor bisa saja user atau suatu fungsi sistem yang menggunakan fungsi atau manfaat yang berada didalam maupun diluar ruang lingkup pada suatu subjek. ▪ Jika actor yang memfungsikan suatu sistem adalah manusia maka dilambangkan sebagai character gambar manusia seperti gambar disamping, jika 	 <p>The diagram illustrates two ways to represent an actor in a Use Case Diagram. The top representation is a stick figure, a common symbol for a human actor, with the text 'Actor/Role' written below it. The bottom representation is a rectangular box containing the stereotype '<<actor>>' and the text 'Actor/Role' below it.</p>
--	---

<p>aktor yang terkait bukan manusia, maka akan dilambangkan sebagai persegi panjang.</p> <ul style="list-style-type: none"> ▪ Tiap aktor dilabeli sebuah peran yang dicatumkan dengan sebuah nama. ▪ Aktor dapat direlasi kan dengan actor lain melalui fungsi <i>specialization/super class association</i>, yang digambarkan dengan sebuah anak panah. ▪ Actor dapat diletakan diruang lingkup suatu sistem. 	
<p>Use Case:</p> <ul style="list-style-type: none"> ▪ Merepresentasikan bagian-bagian pada suatu sistem yang fungsional. ▪ Dapat merelasikan fungsi extend dengan use case lain. ▪ Dapat merelasikan fungsi include dengan use case lain. ▪ Dapat diletakan didalam ruang lingkup suatu sistem. 	

<ul style="list-style-type: none"> Dapat namakan dengan deskripsi kata kerja – frase kata benda. 	
<p>Subjek:</p> <ul style="list-style-type: none"> Penamaan subjek dapat diisi didalam ataupun diatas subjek. Merepresentasikan sebuah ruang lingkup pada suatu sistem yang didalamnya terdapat fungsi proses. 	
<p>Relasi association:</p> <ul style="list-style-type: none"> Berfungsi untuk menghubungkan antara actor dan usecase agar saling berinteraksi. 	
<p>Relasi include:</p> <ul style="list-style-type: none"> Merepresentasikan fungsi relasi include dari antara 1 use case ke use case lainnya. Gambar panah digambarkan dari use case utama menuju use case yang aktif. 	
<p>Relasi extend:</p> <ul style="list-style-type: none"> Merepresentasikan fungsi relasi extend antara 1 use case ke use case lainnya. Gambar panah digambarkan dari 	

use case ke use case utama.	
Relasi generalization: <ul style="list-style-type: none"> ▪ Mempresentasikan use case khusus ke arah yang umum. ▪ Gambar panah digambarkan dari use case khusus ke use case utama. 	

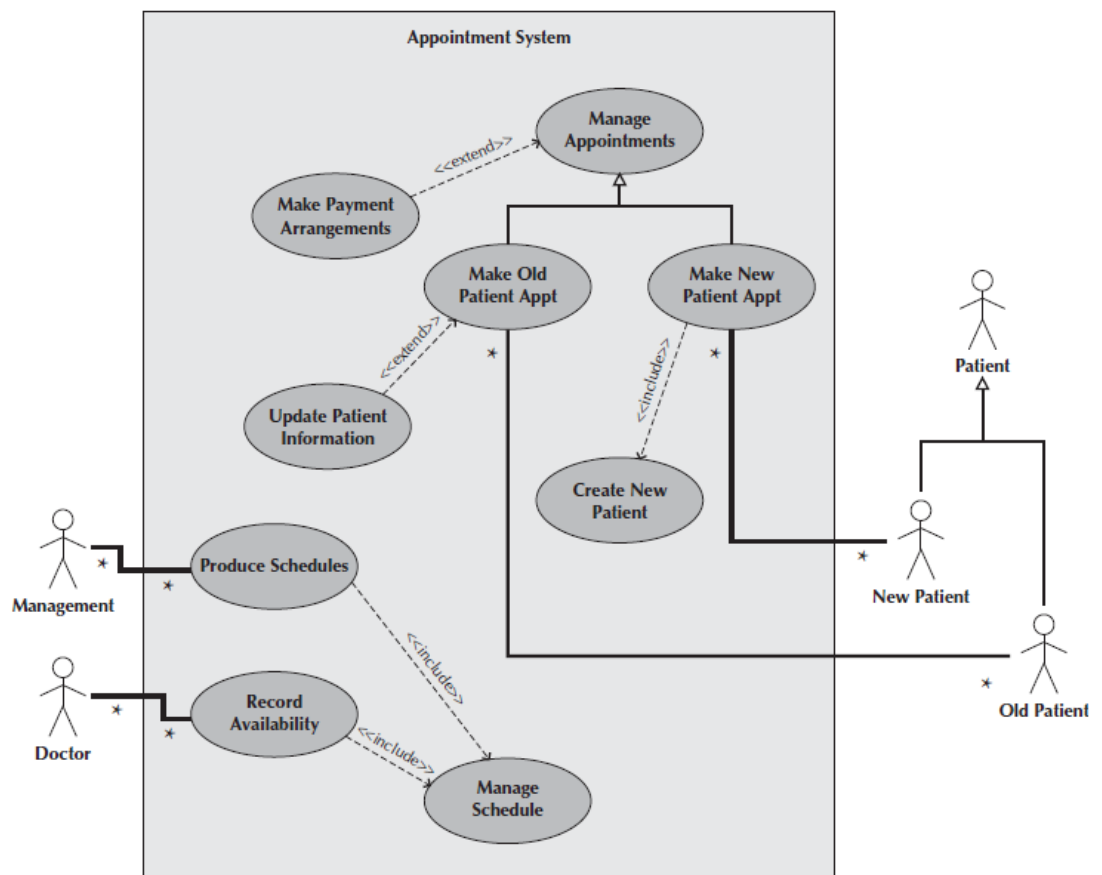
Tabel 11. 1 Tabel Elemen pada Use Case Diagram

Selanjutnya agar mahasiswa lebih memahami dengan tata cara penggunaan use case maka akan dijelaskan lebih lanjut lagi melalui study kasus.

b. Studi Kasus Pada Rumah sakit

Sebelumnya sudah dijelaskan elemen-elemen apa saja yang menjadi bagian penggunaan pada Use Case Diagram ketika merancang suatu sistem oleh karena itu akan dijelaskan tata cara penggunaan use case diagram dengan studi kasus langsung.

Pada Studi kasus ini dikondisikan terkadang suatu aktor dapat memainkan peran khusus dari tipe aktor secara umum. Sebagai contoh, kadang kala pasien baru berinteraksi dengan sistem dengan cara yang agak berbeda dari pasien pada umumnya. Dalam hal ini, aktor khusus(pasienbaru) dapat ditempatkan pada model, yang ditampilkan menggunakan garis dengan segitiga berlubang pada ujungnya.



Gambar 11. 1 Gambar use case pada sistem rumah sakit

Nah Pada Studi kasis ini ada 1 aktor yang mempunyai hak kusus langsung tanpa harus melakukan beberapa prosedur kusus yang dibiasa dilakukan pada aktor umumnya, yaitu pasien tua. Pasien tua ini bisa langsung masuk ruang lingkup sistem tanpa harus memenuhi beberapa aktivitas yang biasa dilakukan pasien lain pada umumnya. Dikarenakan pasien tua memiliki hak kusus yang berbeda pada pasien lainnya. Itulah kenapa adanya suatu aktor dapat kondisikan mempunyai hak kusus dari aktor lain pada umumnya yang mengikuti aturan pada sistem dikarenakan sebuah aktor dapat diperlakukan secara khusus.

Ruang lingkup sistem use case diapit oleh batas subjek, yang dimana sebuah kotak didefinisikan ruang lingkup sistem dan jelas digambarkan didiagram itu. Satu keputusan yang sulit dibuat dikarenakan tempat menggambar pada batas subjek. Batas Subjek ini dapat digunakan untuk memisahkan sistem perangkat lunak dari ruang lingkup sistem, subsistem dari subsistem lain pada perangkat lunak sistem, atau proses individu dalam

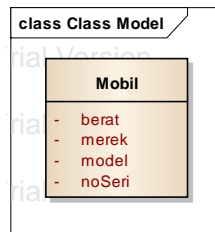
sistem perangkat lunak. Ini juga dapat digunakan sebagai pemisah sistem informasi, termasuk perangkat lunak dan aktor dalam, dari ruang lingkup sistem. Nama Subjek dapat muncul di dalam atau di atas kotak. Subjek batas ditarik berdasarkan ruang lingkup sistem. Dalam sistem penunjukan, diasumsikan bahwa para aktor manajemen dan dokter, berada pada ruang lingkup pada suatu sistem, dikarenakan merekalah yang menggunakan sistem tersebut. Sistem dapat memasukan resepsionis sebagai aktor. Namun pada kasus ini, diasumsikan bahwa resepsionis, adalah aktor dalam yang merupakan bagian dari kasus penggunaan Kelola janji temu yang berinteraksi dengan aktor pasien. Oleh karena itu resepsionis tidak digambarkan pada diagram diatas.

2. Class Diagram

Class itu sendiri yaitu sekumpulan obyek yang dimana setiap obyeknya memiliki atribut dan operasi. Jadi Class Diagram itu sendiri suatu diagram yang dimana didalamnya terbuat dari serangkaian objek, dimana masing-masing objek mempunyai atribut dan operasi. Oleh karena itu akan dijelaskan elemen-elemen yang berada pada tiap-tiap 1 class pada suatu class diagram.

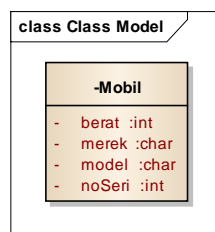
a. Attribute.

Attribute merupakan elemen yang ada pada suatu class. Attribute menggambarkan batas nilai yang mungkin ada pada obyek dalam suatu class. Satu class bisa saja mempunyai kosong atau lebih dari 0 atribut. Secara konvensi, jika nama attribute terdiri dari sebuah suku kata, maka dideskripsikan dengan huruf kecil. Tetapi jika nama attribute mempunyai lebih dari sebuah suku kata maka semua suku kata tergabung dengan suku kata awal memakai huruf kecil dan awal suku kata berikutnya memakai huruf besar. Berikut ini contoh dari atribut yang berada pada suatu mobil.



Gambar 11. 2 Gambar class mobil

Jadi pada suatu mobil mempunyai elemen-elemen atribut pada benda tersebut, pada suatu Class tiap-tiap atribut memiliki type data masing-masing sesuai keinginan siperancang suatu aplikasi ketika ingin mengindentifikasi suatu atribut yang berada pada class diagram.



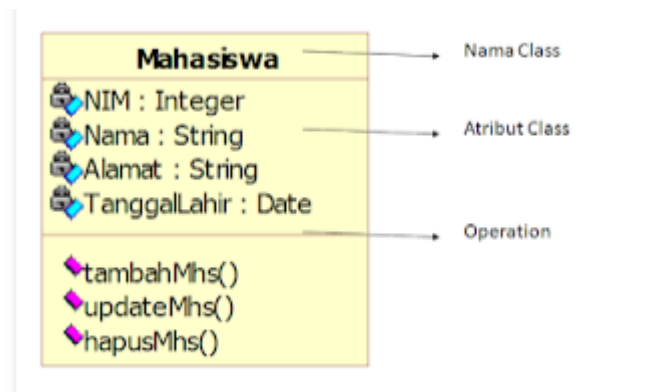
Gambar 11. 3 Class dan type datanya.

Jadi Berat pada class disini diidentifikasi sebagai *integer*, lalu untuk merek sebagai character atau deskripsi, lalu model juga sebagai *character* atau deskripsi, dan terakhir untuk noSeri saya klasifikasikan sebagai *integer*, noSeri ini, sebagai primary key, atau penanda pada suatu mobil memiliki nomor kode produksi unik tersendiri yang berbeda pada setiap mobilnya, meskipun dengan merek model yang sama, tetapi setiap mobil kendaraan yang diproduksi suatu perusahaan memiliki noSeri pembuatan yang berbeda. Hal inilah yang nantinya agar dapat mengindentifikasikan data kedalam database secara mudah.

b. Operation

Operation adalah sesuatu yang hanya dikerjakan pada sebuah class yang hanya dapat dikerjakan oleh class itu sendiri. Setiap class mempunyai fungsi masing-masing ketika melakukan aktivitas. Akan dijelaskan

seperti contoh dibawah ini.



Gambar 11. 4 Class diagram yang memiliki fungsi operation

Yang berada diposisi kotak dibawah adalah sebuah operasi, fungsi ini sangat membantu mendeskripsikan suatu fungsi pada tiap-tiap kelas yang pada nantinya akan memudahkan seseorang ketika merancang suatu aplikasi, dengan deskripsi ini dapat mengetahui bawah fungsi apa saja yang nantinya akan dimasukan pada suatu class, agar memunuhi seluruh kebutuhan yang ada pada suatu class tersebut.

c. Penyederhanaan Class Diagram

Saat Class Diagram terisi penuh dengan semua class dan relasi dengan dunia nyata sistem, class diagram bisa menjadi sulit untuk diartikan atau bisa menjadi lebih komplkes dimengerti oleh orang awam. Jika ini terjadi, terkadang diagram perlu disederhanakan. Namung tergantung pada jumlah asosiasi yang terhubung ke kelas abstrak, jika banyakan maka akan sulit dipahami diagram yang kita buat.

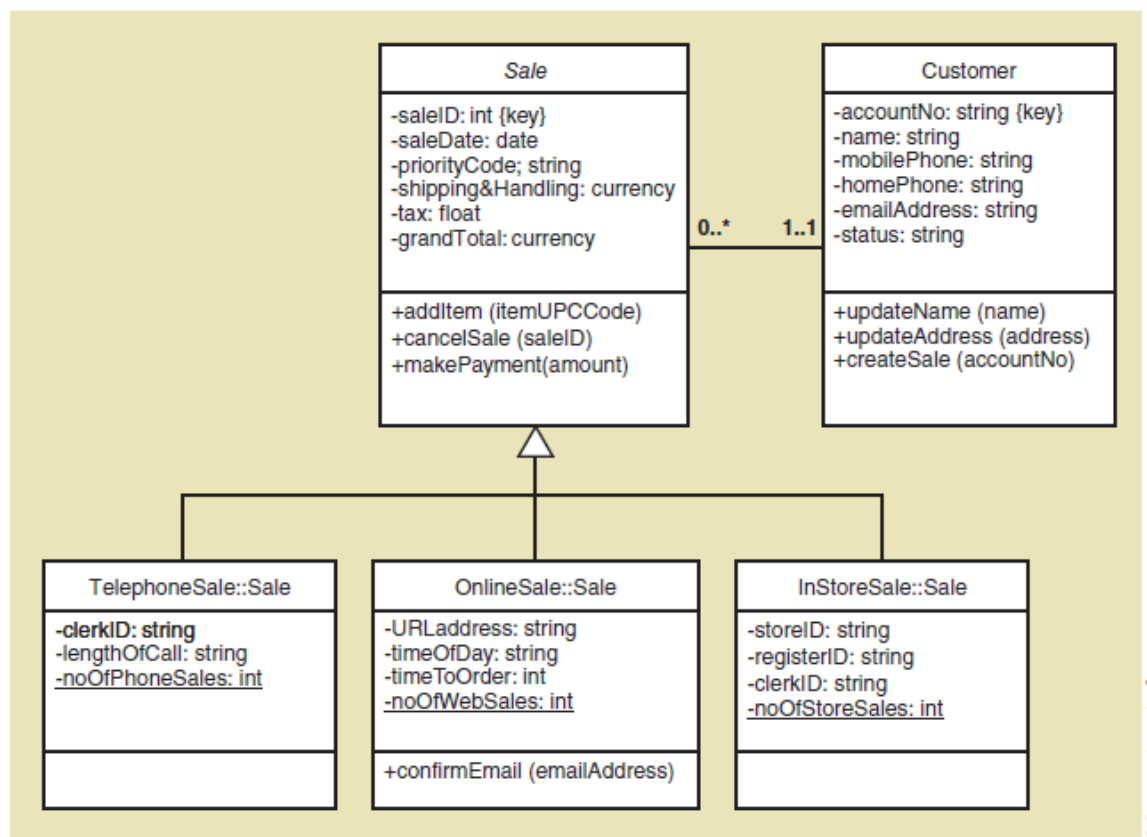
Cara untuk menderhanakan class diagram adalah dengan menggunakan tampilan view. Tampilan awalnya dikembangkan dengan sistem manajemen database relasional untuk ditampilkan saja yang berada pada database. Dalam hal ini, view akan berguna pada class diagram yang menampilkan relasi antar class yang relevan. Pada sudut padang lain akan menunjukan relasi-relasi tertentu, seperti agregasi, asosiasi, atau generalisasi. Setiap kelas, misalnya, hanya menampilkan nama class dan atribut, atau nama class dan operasi.

Care selanjutnya untuk menderhanakan class diagram yaitu dengan menggunakan fungsi *package*. Untuk membuat diagram lebih mudah dibaca

dan menyimpan model-model. Dengan menggunakan *package* tingkat kompleksitas pada suatu class-class akan dikelompokkan menjadi sebuah *package*. *Package* adalah konstruksi umum yang dapat diterapkan ke salah satu elemen dalam model UML.

d. Studi Kasus Class Diagram Pada Sistem Perbelanjaan Online

Berikut ini contoh studi kasus Bentuk class diagram, pada sistem perbelanjaan online dan beserta deskripsi penjelasannya.



Gambar 11. 5 Contoh Class Diagram pemesanan belanja online.

Bisa dilihat pada kolom Class Sale terdapat atribut *saleID*, *saleDate*, *priorityCode*, *shipping&Handling*, *tax*, *grandTotal*. Yang mempunyai operasi, *add item*, *cancelSale*, *makePayment*. Pada Class *costumer* terdapat atribut *account*, *name*, *mobilePhone*, *homePhone*, *emailAddress*, dan *status* dan memiliki operasi *update*, *updateAddress*, *createSale*. Pada *TelephoneSale::Sale* terdapat atribut, *clerkID*, *lengthOfCall*, *noOfPhoneSales*. Pada class *OnlineSale::Sale* terdapat atribut *URLaddress*,

timeOfDay, *timeToOrder*, *noOfWebSales* dan memiliki fungsi operasi *confirmEmail*. Pada class *InStoreSale* terdapat atribut *storeId*, *registerID*, *clerkID*, *noOfStoreSales*.

Dari kesimpulan diatas bahwa tidak selalu setiap class mempunyai sebuah operasi. Dikarenakan Tiap class mempunyai karakteristik masing-masing sesuai kebutuhan sang pembuat aplikasi ketika merancang suatu aplikasi, disini sangat diperlukan analisa yang cukup baik untuk mengidentifikasi fungsi-fungsi tiap class, agar aplikasi nantinya dapat berjalan sesuai dengan permintaan user.

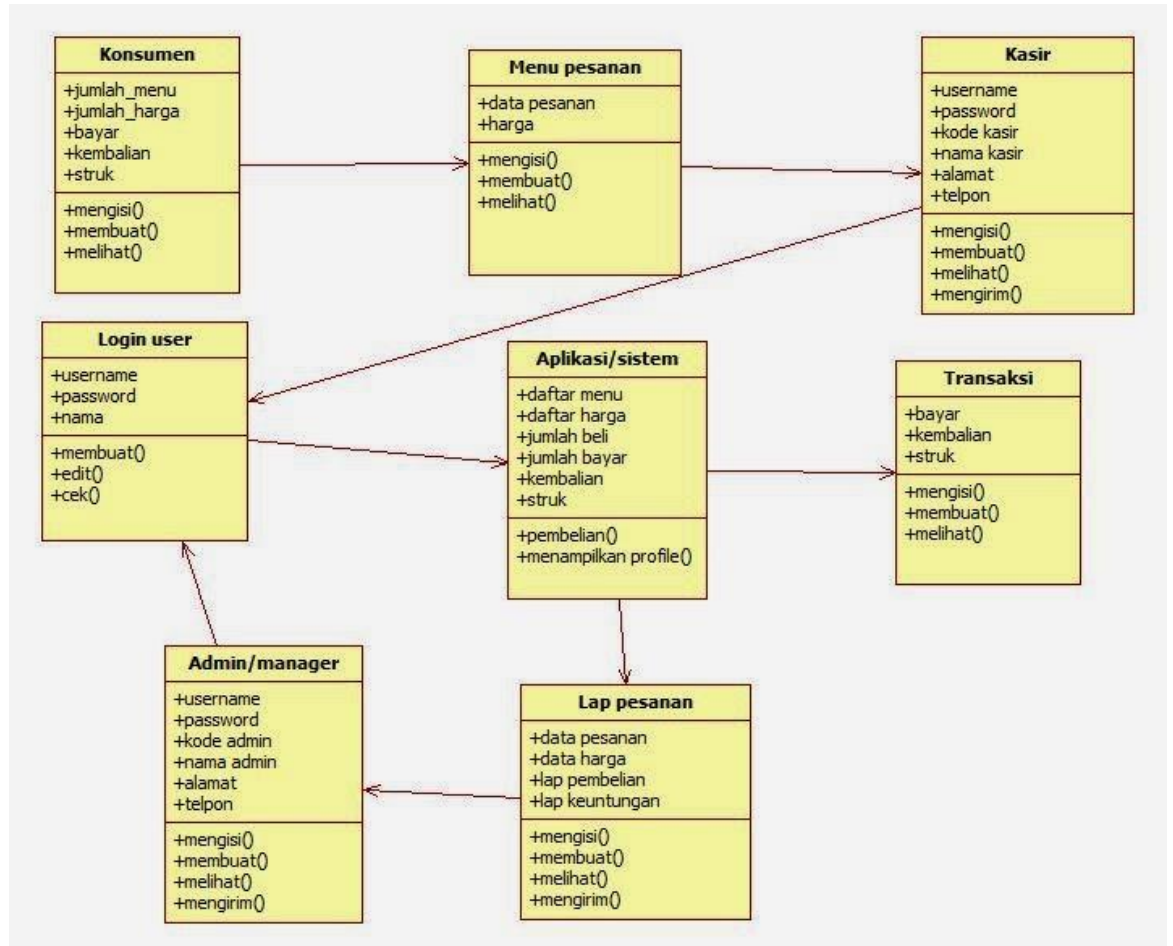
3. Object Diagram

Pembahasan selanjutnya adalah Object Diagram, Object Diagram itu sendiri sebenarnya berasal dari class diagram, hanya saja pada class diagram hanya dijelaskan class, atribut, dan nama operasi secara abstrak, sedangkan pada object diagram menggambarkan obyek-obyek dengan jelas, Dengan menjelaskan dan mendeskripsikan tiap-tiap fungsi pada suatu atribut agar membuat para user dengan mudah memahami suatu sistem aplikasi yang sedang kita buat. Tujuan dibuatnya Object Diagram agar user dapat membandingkan perbedaan yang mewakili class diagram dengan object diagram atau contoh aktivitas yang berada pada setiap elemen-elemen atribut pada Class Diagram. Secara ringkas tujuan dibuatnya Object diagram adalah menunjukan relasi objek dari class ke class yang lain pada suatu sistem, memahami perilaku objek dan hubungan mereka antara 1 class dengan class yang lain.

Meskipun diagram kelas diperlukan untuk mendokumentasikan struktur kelas, sedetik saja. Jenis diagram struktur statis, disebut diagram objek karena berguna untuk mengungkapkan tambahan informasi. Object Diagram pada dasarnya adalah contoh dari semua atau sebagian Class Diagram. Instansiasi berarti berarti membuat instance kelas dengan himpunan yang sesuai nilai atribut. Diagram objek bisa sangat berguna saat mencoba mengungkap detail class. Singkatnya pada object diagram lebih konkret daripada Class diagram yang pengungkapannya masih terbilang abstrak.

Berikut ini salah 1 contoh object diagram pada sistem yang ada pada pemesanan makan di restoran.

a. Studi Kasus Object Diagram Pada Sistem Pemesanan Makanan Di Restoran



Gambar 11. 6 Object Diagram pemesanan makan di restoran

Jadi sudah terlihat jelas perbedaan dari Class Diagram dan Object Diagram, Jika pada Class Diagram kita hanya menjelaskan type elemen pada suatu atribut yang berada didalam class serta perelasianya dengan class yang lain, sedangkan pada Object diagram kita dapat melihat jelas suatu fungsi pada suatu atribut pada tiap classnya serta relasinya yang dapat kita pahami dengan mudah.

Pada Object Diagram, kita dapat menerjemahkan informasi-informasi yang ada pada suatu class, tanpa melihat detail secara abstraksi. Semua aktifitas terdeskripsi sangat jelas beserta operation pada tiap-tiap class.

b. Tujuan Object Diagram

Tujuan Object diagram mirip dengan class diagram. Bedannya, class diagram mewakili model abstrak yang terdiri atas class-class dan hubungannya. Sedangkan Object Diagram merupakan contoh langsung dari Class Diagram. Tujuannya adalah menangkap pandangan statis pada sebuah sistem.

Secara ringkas, tujuan object diagram yaitu Untuk *forward* dan *reverse engineering*, Menunjukkan relasi objek dari suatu sistem, menunjukkan pandangan statis dari suatu interaksi, serta memahami perilaku objek dan hubungan mereka dari perspektif praktis.

c. Penggunaan Object Diagram

Object diagram sangat berguna dalam menampilkan sampel-sampel obyek yang berhubungan antara satu dengan lainnya. Dalam banyak contoh, rangkaian yang tepat dapat dideskripsikan secara tepat menggunakan diagram kelas, tetapi struktur tersebut masih bersifat abstrak dan sulit untuk dipahami. Pada keadaan seperti ini membuat contoh dengan obyek diagram akan mempermudah pengerjaan. Untuk memodelkan sebuah struktur obyek bisa dilakukan dengan langkah-langkah sebagai berikut :

- 1) Identifikasikan mekanisme yang ingin dimodelkan. Sebuah mekanisme mewakili beberapa fungsi atau perilaku dari Sebagian sistem yang akan dimodelkan sebagai hasil dari interaksi dari class, interface dan hal-hal yang lain.
- 2) Untuk setiap mekanisme, identifikasi class-class, tampilan muka dan elemen yang lain yang berpartisipasi pada kolaborasi ini. Selanjutnya perlu diidentifikasi relasi diantara semua elemen tersebut.
- 3) Pertimbangkan satu skenario yang meliputi semua mekanisme tersebut. Pertahankan skenario tersebut sementara waktu dan ulangi untuk setiap obyek yang berpartisipasi pada mekanisme tersebut.
- 4) Ekspose nilai state dan attribute dari setiap obyek untuk memahami skenario tersebut.
- 5) Dengan cara yang sama, ekspose link diantara obyek-obyek yang mewakili instance dari asosiasi antara mereka.

C. SOAL LATIHAN/TUGAS

1. Carilah definisi pengertian Use Case Diagram, Class Diagram, dan Object Diagram selain yang disebutkan diatas !
2. Definisikan pengertian Use Case Diagram, Class Diagram, dan Object Diagram menurut pendapatmu !
3. Buatlah studi kasus contoh untuk pembuatan Use Case Diagram !
4. Buatlah studi kasus contoh untuk pembuatan Class Diagram !
5. Buatlah studi kasus contoh untuk pembuatan Object Diagram !

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

John W. Satzinger, Robert B. Jackson, Stephen D. Burd (2016). *Systems Analysis and Design In A Changing World*. 7th edition. Boston: Cengage Learning.

GLOSARIUM

User adalah istilah lain dari pengguna aplikasi.

Use Case Diagram adalah sebuah gambaran diagram *UML* yang dipakai untuk membuat perancangan sebuah sistem.

Association adalah sebuah fungsi perelasian untuk merelasikan hubungan antara sebuah element dengan element lainnya.

Include adalah sebuah fungsi perelasian yang digunakan ke sebuah use case untuk menjalankan sebuah fungsi.

Extend adalah sebuah fungsi perelasian yang digunakan ke sebuah use case yang nantinya use case tersebut dapat berfungsi tanpa bantuan use case tambahan.

Attribute adalah merupakan elemen atribut pada suatu kelas

Object adalah sebuah objek atau benda/element yang berada pada suatu diagram.

Int merupakan sebuah elemen yang memiliki type data angka.

Char merupakan sebuah elemen yang memiliki type data huruf.

Operation adalah suatu perintah yang dapat dilakukan pada sebuah kelas.

Forward Engineering merupakan perancangan dan implementasi hasil dari reverse engineering untuk menghasilkan sistem yang baru.

Interface adalah tampilan muka atau tampilan yang dapat dilihat oleh pengguna.

PERTEMUAN 12

IMPLEMENTASI DIAGRAM UML PADA STATECHART DIAGRAM DAN ACTIVITY DIAGRAM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian Statechart Diagram, Activity Diagram, elemen-elemen yang ada pada diagram tersebut, dan tata cara pembuatan yang ada pada Diagram tersebut.

B. URAIAN MATERI

1. Statechart Diagram

Statechart Diagram adalah sebuah diagram yang mendeskripsikan memperlihatkan tentang perilaku sistem atau aplikasi yang dibuat dengan penggambaran yang dibuat kotak yang digambarkan sebuah objek, serta adanya penandaan panah yang berfungsi sebagai penunjuk perpindahan alur suatu state ke state lain, tetapi objek tersebut masih bersifat abstrak. Interaksi diagram menunjukkan pesan yang melewati antara obyek-obyek di dalam sistem pada jangka waktu yang pendek. Statechart diagram menyajikan simbol-simbol dan sejumlah ide untuk sebuah pemodelan. Tipe diagram ini berpotensi menjadi sangat kompleks pada waktu yang singkat. Diperlukannya Statechart Diagram untuk mempermudah analisa, pembuat dan pendevelope untuk membaca tingkah laku obyek yang berada disistem.

Dari Statechart Diagram inilah nantinya akan menghasilkan sebuah Activity Diagram, Activity Diagram itu sendiri merupakan use case khusus yang berada pada Statechart Diagram yang menjelaskan gambaran-gambaran aktivitas yang lebih konkrit, daripada yang digambarkan pada Statechart Diagram, yang masih berbentuk begitu abstrak. Meskipun kedua diagram sama-sama menggambarkan perilaku dan aspek yang berada pada suatu sistem, tetapi keduanya mempunyai perbedaan gambar permodelan ketika diagram dibuat. Pada Activity Diagram dipakai untuk pemodelan yang menggambarkan langsung proses bisnis yang dimana beberapa objek

langsung berpartisipasi pada suatu sistem. Sedangkan pada Statechart diagram cenderung digunakan untuk memodelkan siklus riwayat hidup pada sebuah objek reaktif. Sebuah objek reactive menyediakan sebuah konteks untuk suatu statechart. Konteksnya yaitu :




- Respon terhadap *external events* (even atau kejadian yang terjadi diluar konteks objek).
- Mempunyai model siklus hidup yang jelas sebagai progress suatu statem transisim dan *events/kejadian*.
- Memiliki sifat yang tergantung pada sifat sebelumnya.

Sebuah statechart terdiri dari 1 buah state mesin untuk setiap objek reaktif. Pada dunia nyatayang penuh objek reaktif dapat dimodelkan menggunakan state mesin. Pada pemodelan objek orientasi, dapat menggunakan state mesin untuk memodelkan tingkahlaku yang dinamis terhadap objek reaktif seperti *Classes*, *Use Cases*, *subsystems*, sistem keseluruhan. Namun state mesin umumnya digunakan untuk memodelkan tingkah laku sebuah class yang dinamis. Selanjutnya akan dijelaskan elemen-elemen yang ada pada statechart diagram.

a. Elemen pada Statechart Diagram

Dibagian ini akan dijelaskan elemen-elemen apa saja yang dibutuhkan ketika membuat sebuah Statechart Diagram.

<p>Simbol Transisi</p> <p>Transisi merupakan gambaran sebuah panah yang berfungsi sebagai penunjuk arah jalur dari suatu state ke state selanjutnya. Transisi tidak hanya menunjuk kearah state lain, tetapi dapat difungsikan juga untuk menunjuk kearah baliknya atau diri sendiri.</p>	
--	--

<p>Simbol Initial State</p> <p>Initial state biasanya berfungsi sebagai penanda awal berjalannya suatu sistem yang digambarkan pada statechart diagram itu sendiri.</p>	 <p>Initial state</p>
<p>Simbol Final State</p> <p>Simbol ini menunjukkan berakhirnya sebuah aktivitas yang terjadi pada sebuah sistem yang digambarkan pada Statechart Diagram.</p>	 <p>Final state</p>
<p>Simbol State</p> <p>Simbol ini menunjukkan suatu aktivitas yang terjadi pada suatu state yang berada pada Statechart Diagram.</p>	 <p>State</p> <p>A simple state</p>

Tabel 1. Tabel simbol pada Statechart Diagram

Pada table diatas dijelaskan elemen-elemen apa saja yang ada ketika membuat sebuah Statechart Diagram, kedalam sebuah perancangan.

b. Tujuan Perancangan Statechart Diagram

Statechart diagram merupakan sebuah diagram UML yang dipakai untuk memodelkan sifat dinamis yang terjadi didalam sistem. Diagram tersebut mendefinisikan state yang berbeda dari suatu objek selama siklus hidupnya berlangsung dan state ini diubah karena sebuah *events*/kejadian. Baik itu secara eksternal maupun internal.

Statechart diagram, menggambarkan aliran control dari satu state ke state lain. State didefinisikan dengan suatu kondisi dari satu objek dan state tersebut akan berubah ketika terpicu oleh kejadian/*events*. Tujuan terpenting sebuah Statechart Diagram adalah untuk memodelkan keadaan suatu objek dari awal pembuatan hingga dibuang. Berikut ini tujuan utama Statechart Diagram:

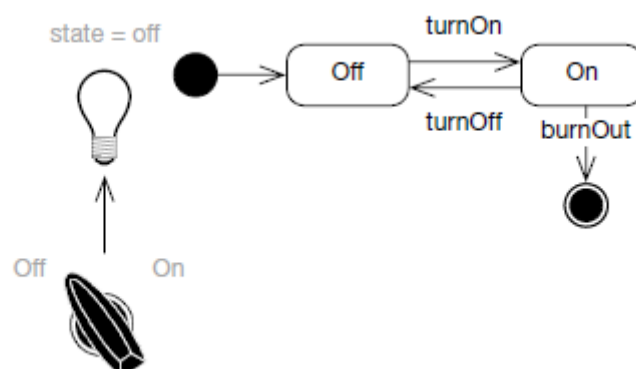
- 1) Untuk memodelkan aspek dinamis yang terjadi pada suatu sistem.
- 2) Untuk memodelkan siklus jangka hidup reaktif yang terjadi pada sistem.
- 3) Untuk mendeskripsikan berbagai status pada suatu state/objek.

Selanjutnya akan dijelaskan studi kasus yang terjadi didunia nyata tentang tata cara penggambaran statechart diagram dengan menggunakan contoh sebuah proses sistem yang terjadi disuatu aktivitas didunia nyata.

c. Contoh state Mesin dan kelasnya

Untuk sebuah class terdiri dari satu state machine semua modelnya ditransisikan antara states terhadap seluruh objek pada suatu class yang merespon sebuah kejadian/*events*. *Events* itu sendiri adalah sebuah pesan yang dikirim oleh objek lain, meskipun dibeberapa kasus objek mungkin terhubung dan menghasilkan suatu kejadian sebagai respon terhadap waktu.

Object yang diberikan pada class dapat berpartisipasi dibanyak use case, dengan fungsi tersebut dapat melihat state mesin untuk sebuah model class dari tingkah laku objek yang berada dikeseluruhan use case. Berikut salah satu contoh pemodelan state mesin simple yang berada disuatu aktifitas kejadian didunia nyata. Contoh ketika menyalakan lampu



Gambar 3. Contoh state mesin menyalakan lampu bohlam

Setiap state mesin diawali dengan simbol initial state (lingkaran tebal) yang mengindikasikan berawalnya sebuah mesin, dan terhenti disimbol lingkaran yang keseluruhan titik tidak tebal, yang mengindikasikan state suatu mesin berakhir. Ketika tombol switch dinyalakan maka akan menjalankan suatu kejadian/*events* yang kalo dibahasakan dengan *pseudo-state* akan menjalankan state event *turnOn*. Pada suatu state mesin, kejadian dipertimbangkan secara seketika. Di lain kata, hanya membutuhkan waktu kosong untuk suatu kejadian yang dikirim dari sakelar ke lampu bola. Kejadian seketika memberikan penyederhanaan yang penting untuk teori state mesin yang membuatnya lebih gampang berinteraksi atau menurut. Tanpa kejadian seketika kita akan melakukannya dengan beberapa kondisi yang besaingan, Yang dimana dua kejadian bersaing dari sumber untuk mencapai objek reaktif yang sama. Dibutuhkan model persaingan kondisi pada sebuah state mesin.

Bola lampu menerima event *turnOn* dan state berubah menjadi *On* ketika merespon sebuah kejadian/*events*. Ketika menerima kejadian state *turnoff* yang dikirimkan ke bola lampu, maka state berubah menjadi *off* dan lampu menjadi padam.

Pada point ini, kejadian *burnout* mungkin terjadi, dan dan menyelesaikan state mesin.

d. State

Pada sebuah UML state didefinisikan sebagai, sebuah kondisi atau situasi yang terjadi ketika hidupnya sebuah objek yang dimana tertarik dengan sebuah kondisi, menjalankan sebuah aktivitas, atau menunggu suatu kejadian/*event*. Sebuah state pada objek bervariasi dari waktu ke waktu, tetapi titik tertentu ditentukan oleh :

- Nilai atribut pada suatu objek.
- Hubungannya terhadap objek lain.
- Aktifitas yang dijalankannya.

Namun, dari sudut pandang penggunaan terhadap bola lampu, state yang dipakai untuk membedakannya hanya ada state *On* dan *Off*. Ini adalah suatu hal penting kunci kesuksesan untuk membuat pemodelan

Statechart Diagram. Sebagai contoh dibawah

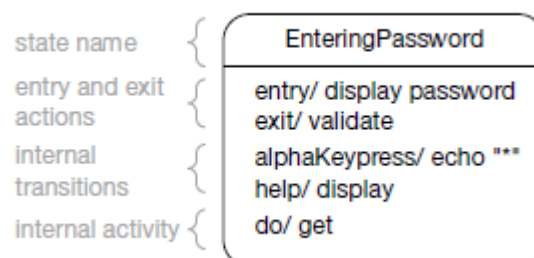
```
class Color
{
    int red;
    int green;
    int blue;
}
```

Gambar 4. Contoh penggambaran nilai atribut.

Seandainya kita asumsikan jika red, green dan blue masing-masing dapat menerima nilai antara 0-225 lalu, yang hanya berdasarkan dari nilai atributnya, berarti tiap objek pada suatu class mempunyai 256 kemungkinan jika dikalikan ke tiganya $256 \times 256 \times 256 = 16777216$ kemungkinan state yang keluar dan itu akan menjadi beberapa statechart. Namun, secara fundamental bermunculan pertanyaan pada diri sendiri secara fundamental. Apa kunci semantic perbedaan antara tiap-tiap state? Tentu saja jawabannya tidak ada. Tiap 16777216 kemungkinan state hanya merepresentasikan tiap warna, hanya itu saja. Faktanya, statechart untuk tiap classnya menjadi membosankan, seperti yang dilihat hanya kemungkinan tiap state saja yang muncul tanpa menjelaskan proses sistem apa yang sedang dilakukan. Singkatnya harus ada perbedaan yang membuat perbedaan semantic antar state untuk memodelkannya pada suatu state mesin.

e. State Syntax

Berikut ini sebuah gambaran state syntax disini akan dijelaskan secara mendetail elemen-elemen apa saja yang ada tiap suatu state ketika merancang suatu state diagram.



Gambar 5. Contoh state ketika memasukan suatu password

Entering password untuk nama state, entry/display password dan

exit/validate perintah untuk menginput dan keluar, alphaKeypress/echo dan help/display merupakan transisi internal yang terjadi didalam sistem, do/get merupakan aktifitas internal yang terjadi pada suatu state. Tiap sebuah state dijelaskan secara abstrak fungsi-fungsi apa saja yang terjadi ketika kita menginput sebuah password.

Tiap state terdiri dari nol tindakan atau lebih dan sebuah aktifitas. Tindakan terbagi menjadi seketika dan tak dapat terganggu, sedangkan aktifitas membutuhkan waktu dan dapat diganggu. Setiap aksi pada state diasosiasikan dengan transisi internal yang memicunya sebuah kejadian atau *event*. Bisa berupa beberapa nomor aksi dan internal transisi didalam suatu state.

f. Transitions

Transisi mempunyai syntax yang simple yang digunakan untuk menggunakan fungsi *external transitions* (yang ditunjukan dengan arah panah) ataupun *internal transitions*. Tiap transisi mempunyai tiga opsi elemen:

- 1) Sebuah kejadian – Sebuah kejadian internal atau eksternal yang memicu transisi.
- 2) Kondisi bertahan – Sebuah ekspresi Boolean yang harus bernilai “true” sebelum transisi terjadi.
- 3) Sebuah tindakan – sebuah lembar kerja yang mengasosiasikan dengan sebuah transisi, dan terjadi ketika transisi menyala.

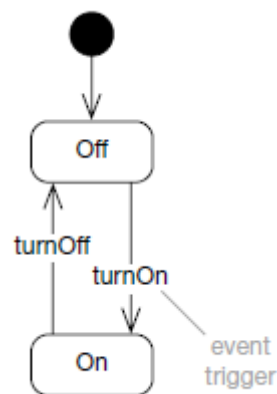
Berikut gambaran sederhana transisi yang terjadi pada suatu state diagram.



Gambar 6. Gambar sebuah transisi pada Statechart

g. Event/Kejadian

Pendefinisian *event*/kejadian pada uml, dispesifikasikan sebagai suatu hal yang patut diperhatikan yang memiliki lokasi didalam ruang dan waktu. *Event*/kejadian memicu transisi yang ada distate mesin. Event ditunjukan diluar pada sebuah transisi. Berikut akan dijelaskan pada gambar.

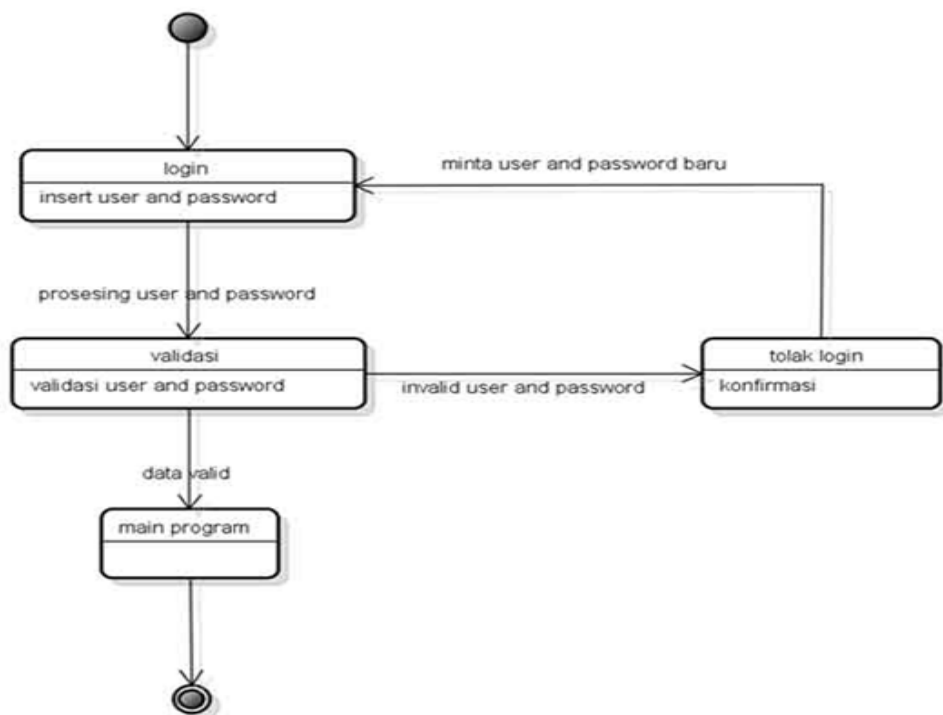


Gambar 7. Gambar event pada suatu state mesin

Dijelaskan pada gambar diatas yang memicu perubahan suatu state dikarenakan adanya suatu transisi yang dapat menimbulkan perubahan status yang terjadi pada suatu state, yang sebelumnya state berstatus off, kini terpicu oleh sebuah transisi “turnOn” menjadi On.

h. Studi Kasus Statechart Diagram

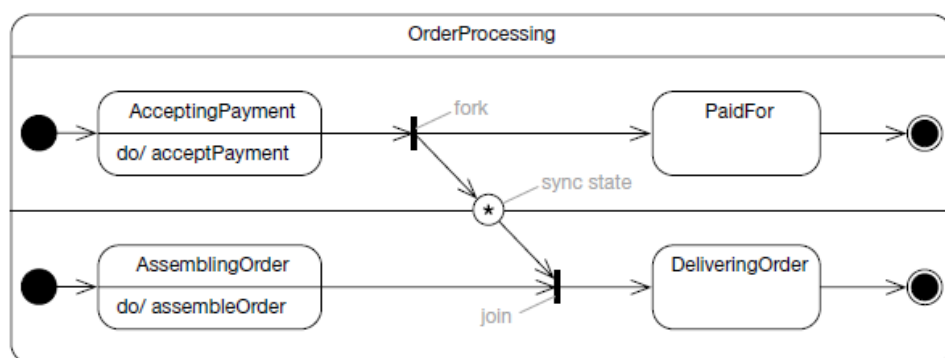
Berikut akan ada contoh studi kasus yang terjadi pada suatu sistem pada fungsi state login, akan dijelaskan elemen-elemen apa saja yang terjadi dibalik adanya fungsi login kedalam suatu sistem.



Gambar 8. Statechart diagram untuk melakukan login pada aplikasi.

Bisa dilihat pada statechart diagram, menggambarkan tahap-tahap suatu aktivitas pada sebuah aplikasi ketika ingin menjalankan fungsi yang ada pada aplikasi. Dari sebuah fungsi login dijelaskan bahwa ketika ingin melakukan login diharuskan mempunyai username dan password yang valid, ketika password kita tidak sesuai maka yang terjadi sistem akan menagih kembali password yang harus dimasukan sampai benar jika tidak akses akan ditolak dan sistem akan kembali kehalaman login utama. Selanjutnya jika berhasil masuk maka sistem akan berjalan dan mempersilahkan user untuk mengakses menu utama.

Selanjutnya akan dijelaskan studi kasus pengkomunikasian penggunaan state yang syncron terhadap dua state yang berbeda tetapi salih berhubungan berikut contoh penggambaran yang terjadi pada sistem pemesanan barang.



Gambar 9. Statechart diagram pada sistem pemesanan

Kedua statechart tersebut berbeda dan mempunyai tugas masing-masing tetapi saling terhubung satu sama lain, ketika sistem menerima pembayaran selanjutnya akan menuju ke state berikutnya yaitu kepembayaran untuk kemudian diakhiri dengan final state, Disitu sudah jelas pembayaran sudah dibayarkan untuk kebutuhan konsumen yang telah dipesan. Lalu state yang dibawah ada state merancang pemesanan, pesanan akan dirancang ketika pembayaran telah berhasil dilakukan, jika tidak proses tidak dapat dilakukan. Digambar diatas terlihat ada hubungan *sync state* yang berfungsi mensinkronisasikan suatu perintah ke perintah lain distate yang berbeda, selanjutnya transisi tersebut tersinkron dengan state yang ada Digambar bawah karena saling terhubung, dan kebutuhan sudah memenuhi, makan selanjutnya ke state berikut terjadilah pengiriman

barang, dan diakhiri dengan final state .

2. Activity Diagram

Activity Diagram adalah bagian terpenting pada UML, yang menjelaskan aspek yang mudah dipahami dari sistem dan menggambarkan suatu gambaran aktivitas pada sistem yang jelas membuat penggunaanya lebih memahami alur fungsi pada suatu rancangan aplikasi yang ingin dirancang. Elemen yang digunakan pun sama seperti yang ada pada statechart diagram hanya saja gambaran aktivitas pada Activity Diagram lebih diperjelas. Tujuan dibuatnya adalah untuk menangkap perilaku dinamis yang terjadi pada sistem dengan cara menunjukkan aliran deskripsi dari satu aktifitas ke aktifitas lainnya. Secara umum tujuan pembuatan Activity Diagram dapat dijelaskan sebagai berikut:

- a. Menggambarkan alur aktifitas pada sistem
- b. Menggambarkan urutan aktivitas dari awal sampai akhir/dari satu aktifitas ke aktifitas lainnya.
- c. Menggambarkan percabangan aktifitas yang terjadi pada suatu sistem.

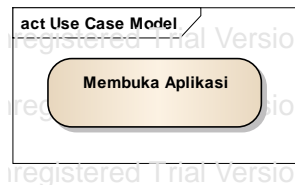
Activity Diagram dapat dilampirkan kesemua elemen permodelan apapun tujuan permodelan dan perilaku elemen tersebut. Activity Diagram dapat dilampirkan kedalam:

- a. Use Case;
- b. Class;
- c. Tampilan atau interface;
- d. Komponen;
- e. Node;
- f. Collaborasi atau diagram campuran;
- g. Operasi dan metode.

Akan sangat diuntungkan ketika menggunakan Activity Diagram untuk proses pemodelan bisnis dan workflow. Umumnya Activity Diagram adalah sebuah flowchart operasi, perlu dipertimbangkan bahwa sebenarnya sumber kode digunakan untuk mengoperasikan. Yang nantinya semua aktifitas tersebut akan dipresentasikan secara lengkap dan ringkas melalui Activity Diagram.

a. Action State

Activity Diagram memiliki action state yang dimana state ini mendeskripsikan suatu proses yang sedang dilakukan distate tersebut. Digambarkan persegi Panjang. Berikut contoh sebuah state pada Activity diagram.

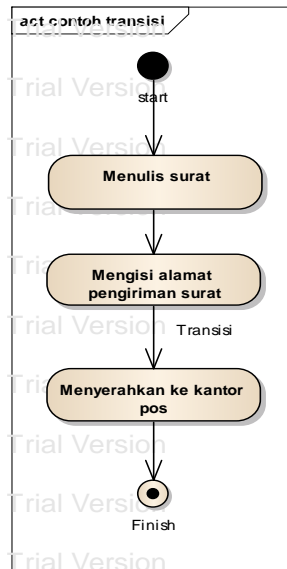


Gambar 29. Gambar sebuah Action state

Pada Activity Diagram dideskripsikan sebuah proses bisnis yang terjadi pada rancangan aplikasi, untuk mempermudah memahami hal-hal apa saja yang terjadi pada suatu state yang berada didalam ruang lingkup sistem. Menurut Spesifikasi UML Action State hanyalah versi penyederhanaan yang lebih umum terhadap elemen yang seringkali disebut sebagai state. Faktanya aksi tersebut dituliskan secara singkat yang mempunyai sebuah aksi dan setidaknya mempunyai transisi kearah luar.

b. Transisi

Sebelumnya dijelaskan seperti apa suatu state yang ada pada elemen Activity Diagram, selanjutnya akan dijelaskan perpindahan-perpindahan aktifitas atau transisi yang terjadi pada suatu Activity Diagram yang terlihat Digambar ini.

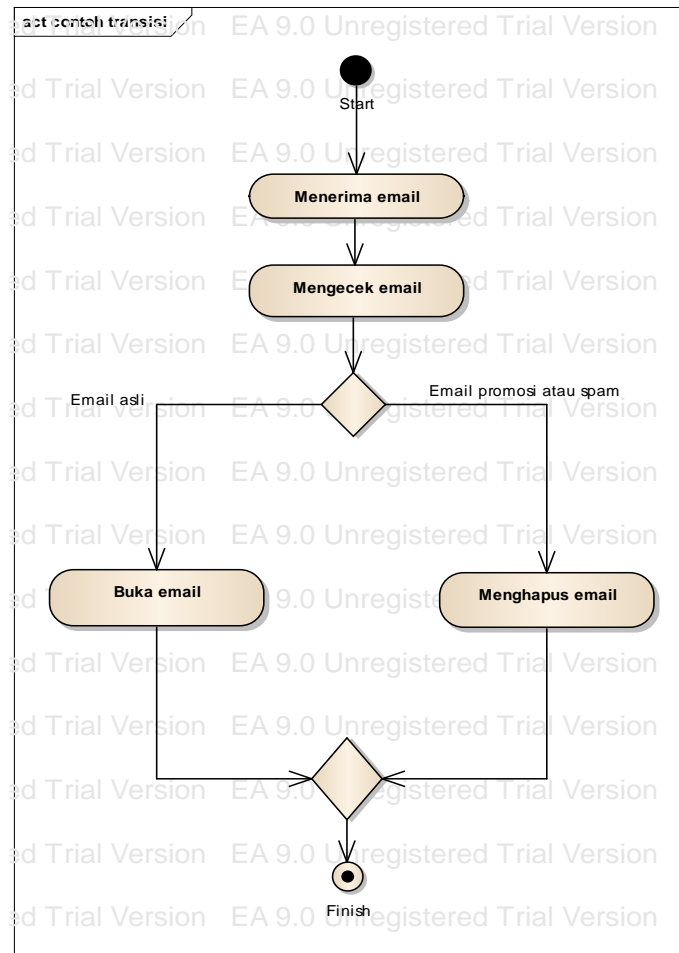


Gambar 30. Contoh perpindahan antara sebuah state ke state yang lain.

Gambar diatas menunjukan alur perpindahan sebuah aktivitas yang terjadi ketika ingin mengirimkan sebuah surat, yang diawal dengan state inisial *start* dan diakhiri dengan state final *finish*. Arah panah menunjukan sebuah transisi atau alur perpindahan sebuah proses dari satu state ke state lainnya.

c. Decision/keputusan

Kali ini akan dijelaskan tentang cara penggunaan fungsi decision yang ada pada activity diagram, diperlukan jika suatu sistem mempunyai lebih dari satu state pada kondisi tertentu, untuk mengelompokan sebuah keputusan. Studi kasus kali ini mengambil contoh dari gambaran aktifitas ketika kita membuka sebuah pesan *email*.



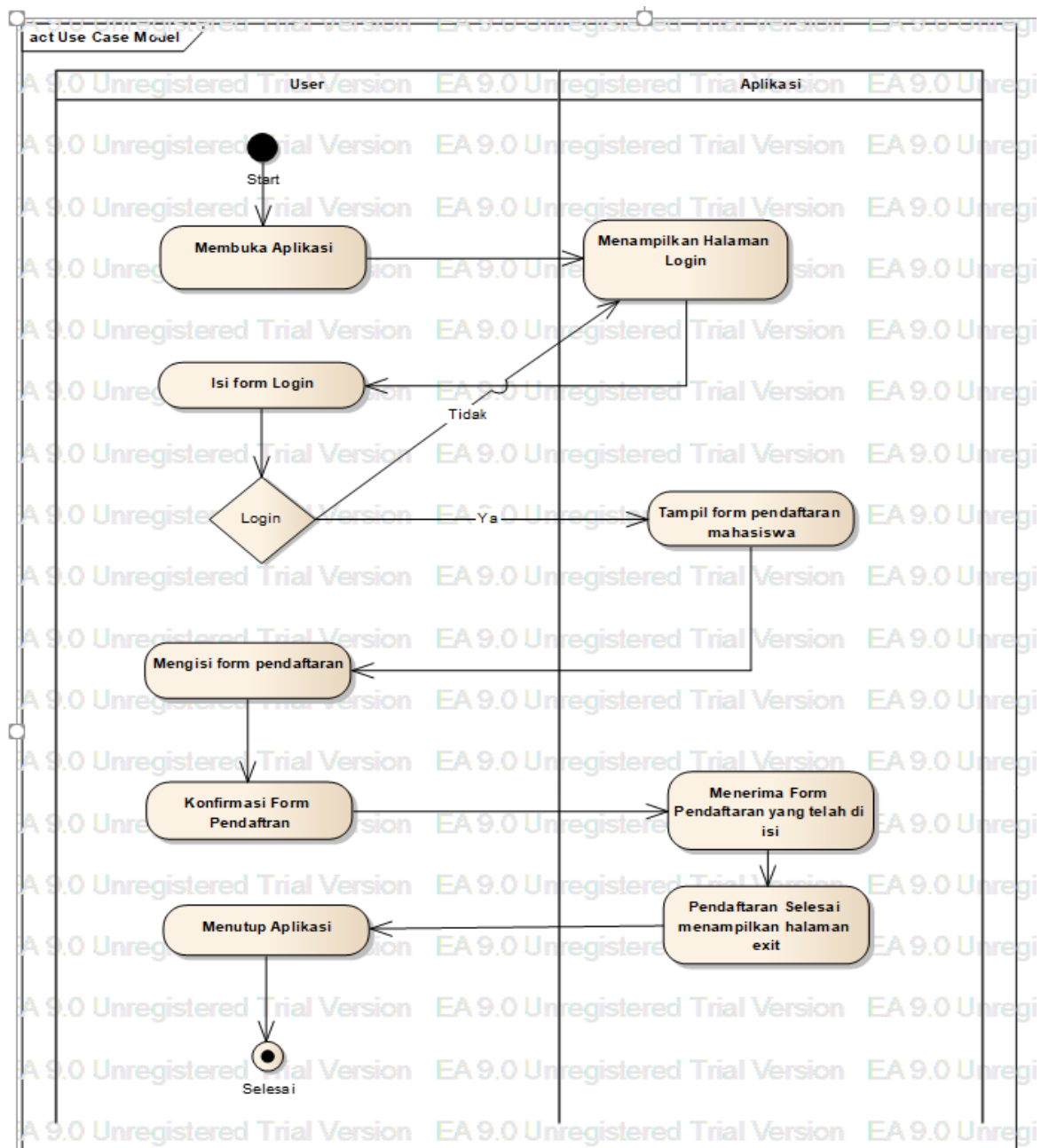
Gambar 31. Contoh penggunaan fungsi decision pada pembuatan Activity Diagram

Jika kita lihat terdapat simbol belah ketupat yang menandakan adanya keputusan yang hanya dapat diambil salah satu ketika sebuah sistem berjalan, seperti gambar diatas ketika membuka email, jika pesan baru yang didapat adalah surat asli, maka email akan dibuka, ketika email atau pesan baru yang didapat ternyata hanya iklan promosi atau spam, maka keputusan yang dibuat email akan diabaikan dan dihapus. Inilah sebuah gambaran fungsi decision yang terjadi didalam suatu Activity

Diagram.

d. Contoh Studi Activity Diagram

Berikut ini sebuah contoh studi kasus Activity Diagram pada sistem pendaftaran mahasiswa baru.



Gambar 32. Activity Diagram penerimaan mahasiswa baru

Pada Activity diatas, state diklasifikasikan antara pengguna atau user dengan aplikasi tersebut. Ini berfungsi untuk mengidentifikasi perbedaan ruanglingkup aktivitas-aktivitas yang terjadi pada tiap state. Adanya pengulangan state dikarenakan kebutuhan yang perlu diinput belum memenuhi persyaratan, sehingga data yang akan masuk nantinya sudah terfilter oleh sistem dan otomatis semua data yang masuk pasti sudah memenuhi kualifikasi kebutuhan pada suatu sistem.

C. SOAL LATIHAN/TUGAS

1. Carilah Definisi pengertian Statechart Diagram dan Activity Diagram selain yang disebutkan diatas!
2. Jelaskan Perbedaan yang ada pada Statechart Diagram dan Activity Diagram menurut kalian!
3. Buatlah Studi kasus contoh untuk pembuatan Statechart Diagram!
4. Buatlah Studi kasus contoh untuk pembuatan Activity Diagram!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc

Jim Arlow, Ila Neustadt (2002). *UML And The Unified Process Practical Object-Oriented Analysis & Design*. Great Britain: Pantek Arts,Ltd, Maidstone, Kent

GLOSARIUM

Events merupakan sebuah kejadian atau peristiwa yang terjadi didalam kelas pada suatu sistem yang berjalan.

Statechart Diagram merupakan diagram *UML* yang dipakai pada salah satu fungsi perancangan pada sistem.

External Events merupakan sebuah kejadian atau peristiwa yang terjadi diruang lingkup suatu sistem tetapi memiliki keterkaitan pada sistem.

Class merupakan sebuah kelas dan salah satu elemen yang ada pada diagram *UML*.

State merupakan sebuah pernyataan pada suatu elemen pada diagram *UML*.

Pseudo-state merupakan bahasa Pseudocode yang menggambarkan algoritma pernyataan.

Pseudocode merupakan metode penulisan yang memakai bahasa sederhana, untuk menjelaskan penggambaran sebuah algoritma.

TurnOn merupakan sebuah contoh pseudo-state yang ada pada suatu sistem.

TurnOff merupakan sebuah contoh pseudo-state yang ada pada suatu sistem.

Activity Diagram merupakan diagram *UML* yang digunakan sebagai perancangan sistem.

Initial State adalah sebuah state yang menggambarkan awal bermulanya berjalannya pada suatu sistem.

Finish State adalah sebuah state yang menggambarkan berakhirnya alur perjalanan pada suatu sistem.

Decision State adalah sebuah state yang menggambarkan suatu sistem harus mengambil sebuah keputusan yang harus diambil ketika memiliki kondisi tertentu.

PERTEMUAN 13

IMPLEMENTASI PERANCANGAN UML LANJUTAN SEQUENCE DIAGRAM, COLLABORATION DIAGRAM, COMPONENT DIAGRAM, DEPLOYMENT DIAGRAM

A. TUJUAN PEMBELAJARAN

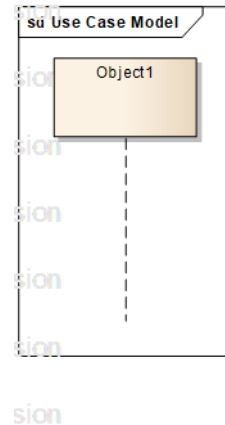
Pada pertemuan ini dijelaskan tentang pengertian dan tata cara merancang sebuah diagram UML lanjutan Sequence Diagram, Collaboration Diagram, Component Diagram, Deployment Diagram.

B. URAIAN MATERI

1. Sequence Diagram

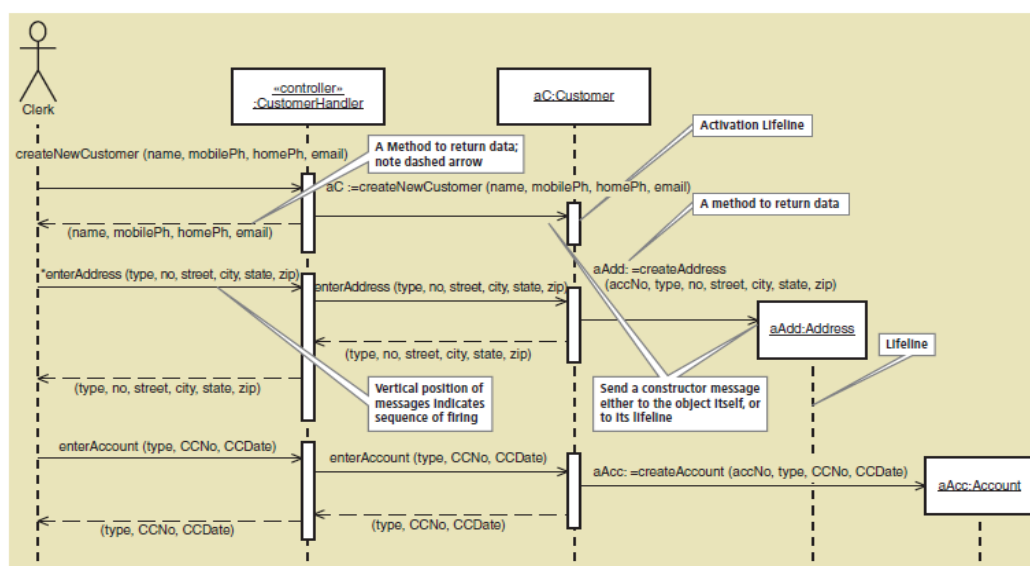
Sequence Diagram digunakan untuk menggambarkan tingkah laku yang berada di skenario. Diagram ini menjelaskan beberapa sampel objek dan pesan yang terletak antara objek yang berada di sebuah use case. Elemen inti sequence diagram biasa dideskripsikan dengan warna kotak persegiempat bernama. Setiap pesan atau *Message* terwakili dengan sebuah garis tanda panah. Objek diletakkan dekat bagian atas diagram mengurut dari kiri ke arah kanan.

Sequence Diagram menunjukkan interaksi objek yang diatur dalam urutan waktu. Sequence Diagram menunjukkan sedikit perbedaan dari yang ada pada Collaboration Diagram di sebuah diagram Objek orientasi. Dimana disini secara signifikan Sequence Diagram menjelaskan secara abstrak tentang apa saja yang terjadi dibalik suatu proses yang terjadi secara berurutan. Sequence diagram mempunyai kemiripan dengan Collaboration Diagram. Mempunyai pandangan yang berbeda tapi mempunyai pemodelan diagram yang sama. Berikut ini sebuah contoh gambar participant yang ada pada Sequence Diagram.



Gambar 33. sebuah participant pada sequence diagram

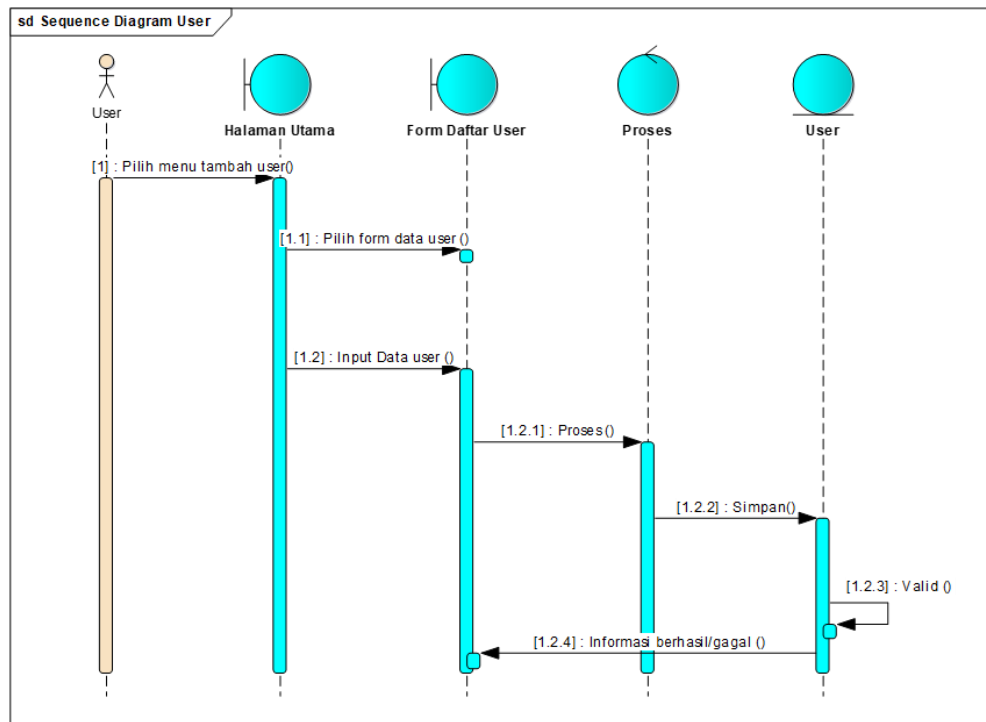
Lalu adanya penandaan aktifitas yang dilambangkan seperti panah, panah tersebut bergerak berurutan dari satu participant ke participant lain dan dari satu lifeline ke lifeline yang lain. Sebuah participant memungkinkan untuk mengirim sebuah *Message* kepada diri sendiri. Penandaan panah tersebut ada 3 macam yaitu, *simple*, *synchronous*, *asynchronous*. Untuk Message simple perpindahan control dari 1 participant ke participant lain. Jika 1 participant mengirim message *synchronous*, maka jawaban ditunggu dan diproses oleh sistem dengan urusan aktivitasnya. Jika yang dikirimkan adalah message *asynchronous*, maka jawaban message tak perlu ditunggu. Sequence Diagram ini bertujuan untuk menunjukkan urutan waktu pesan dari suatu obyek ke obyek lain. Berikut salah 1 contoh sequence diagram untuk pembuatan akun customer.



Gambar 34. Contoh Sequence diagram pembuatan akun costumer

Bisa dilihat ada sebuah aktor, dan 4 partisipan didalamnya, pada sequence diagram dijelaskan detail proses apa saja yang terjadi disetiap participant. Seperti yang ada digambar pembuatan baru akun costumer, penginputan data kelengkapan diri, teridentifikasi sebagai sebuah aktifitas yang terjadi pada sequence diagram dan berurutan tiap prosesnya.

Selanjutnya akan ada studi kasus yang akan dijelaskan pada gambar dibawah



Gambar 35. Sequence diagram menu registrasi akun login

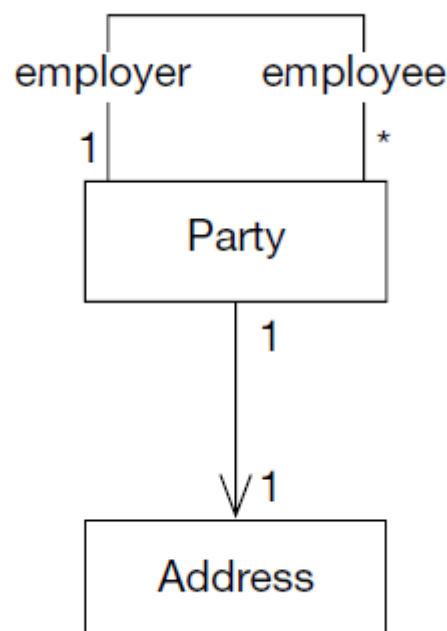
Bisa dilihat participant pada gambar diatas dapat digantikan dengan simbol-simbol lingkaran. Gambar menunjukan tiap-tiap urutan proses participant terdeskripsi dan tiap proses diurutkan melalui panah-panah. Panah tersebut pada akhirnya akan mencapai titik finish yang menandakan berakhirnya sebuah proses yang terjadi pada sistem tersebut.

Disimpulkan bahwa sequence diagram merupakan elemen terpenting dari proses analisa. Dapat mengijinkan perancang diagram untuk melakukan percobaan dan teori-teori serta pendemonstrasian sebagaimana sebuah class dapat berinteraksi dan mengirim secara spesifik perilaku yang berada pada sistem.

2. Collaboration Diagram

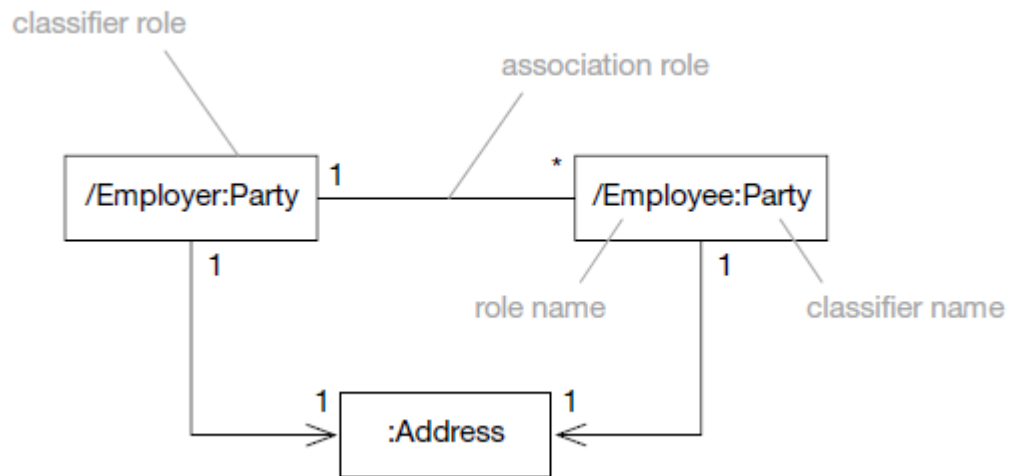
Collaboration diagram digunakan untuk menunjukkan bagaimana suatu obyek berinteraksi untuk menjalankan tingkah laku use case tertentu. Hampir sama pemodelan yang ada pada Sequence diagram tetapi mempunyai perilaku yang berbeda. Pada Collaboration Diagram alur-alur berjalannya sebuah aktifitas digambarkan secara umum atau mudah dimengerti disertai dengan deskripsi lapangan yang terjadi dalam penggunaan suatu sistem tersebut. Tidak seperti Sequence Diagram yang menjelaskan tiap-tiap aktivitas secara abstrak pada suatu sistem tetapi pada Collaboration Diagram berfungsi menjelaskan kepada user bagaimana alur-alur berjalannya aktivitas yang sedang terjadi pada sistem tersebut. Collaboration Diagram berfokus terhadap aspek struktur yang ada pada interaksi objek. Collaborasi diagram memiliki dua bentuk sebagai *descriptor form*, dan *instance form*. *Descriptor form* memberikan pandangan yang umum terhadap *collaboration* dengan menentukan peran yang dimainkan oleh peristiwa atau bisa disebut sebagai pengklasifikasian sebuah peran.

Berikut ini contoh rupa bentuk *descriptor form* yang ada pada Collaboration Diagram.



Gambar 36. Contoh rupa *descriptor form* pada Collaboration Diagram

Kemudian *descriptor form collaboration diagram* bermodelkan seperti gambar dibawah.

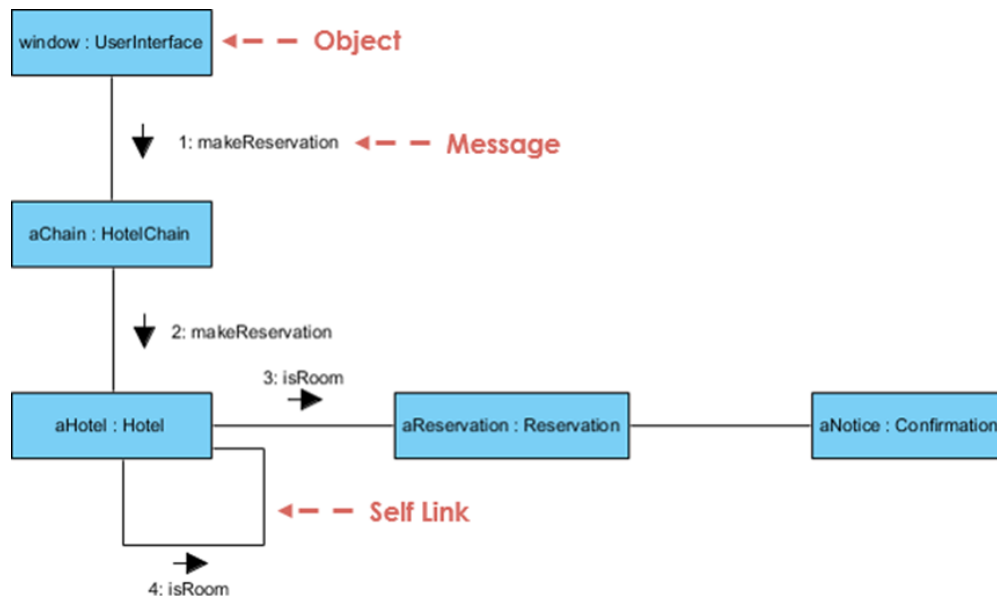


Gambar 37. Descriptor form Collaboration Diagram

Diagram ini menunjukkan peran pengklasifikasi dan peran asosiasi/*association*. Peran pengklasifikasi adalah peran yang dimainkan oleh *instance* dari pengklasifikasi, dan peran asosiasi mengindikasikan peran yang dapat dimainkan oleh *instance* pada asosiasi.

Meskipun diagram ini berisi banyak informasi yang sama dengan Class Diagram, Diagram ini menekankan berbagai peran yang dimainkan oleh *instance* di tiap classnya.

Berikut ini akan dijelaskan studi kasus Collaboration diagram pada sistem reservasi/pemesanan kamar hotel.



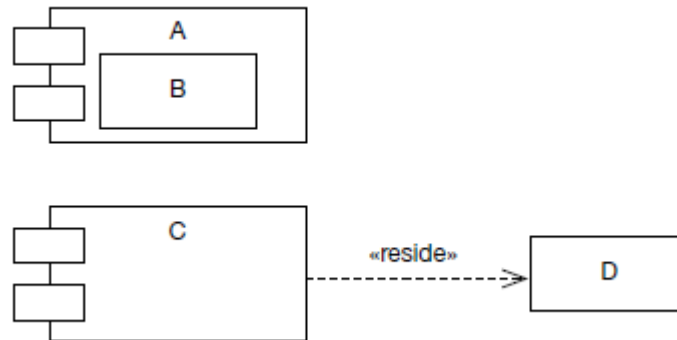
Gambar 38. Contoh Collaboration Diagram sistem pemesanan kamar hotel

Terlihat pada gambar diatas alur-alur pergerakan aktivitas dijelaskan langsung dengan keadaan aslinya, yang sebelumnya pada Sequence Diagram hanya dijelaskan apa yang terjadi dibalik satu aktivitas secara abstrak, kini pada collaboration diagram dijelaskan secara menyeluruh pergerakan aktifitas sampai ke berakhirnya suatu aktifitas pada sebuah diagram.

3. Component Diagram

Component diagram digunakan dalam permodelan aspek bentuk dari sistem yang berorientasi objek yang digunakan untuk memvisualisasikan atau menggambarkan, menentukan, dan mendokumentasikan sistem berbasis komponen dan juga membangun sistem yang dapat dijalankan melalui rekayasa. Component Diagram pada dasarnya adalah class diagram yang berfokus pada komponen sistem yang sering digunakan untuk memodelkan tampilan implementasi secara statis pada suatu sistem. Elemen-elemen yang ada pada sebuah sistem seperti contoh, *sourcefile*, *implementation subsystems*, *AactiveX controls*, *JavaBeans*, *Enterprise JavaBeans*, *Java servlets*, *Java Server Pages*, Elemen tersebut dibuatkan sebuah diagram dan juga mendeskripsikan alur-alur aktivitas yang terjadi didalam suatu sistem dari awal hingga akhir.

Pada dasarnya component diagram adalah sebuah diagram yang menjelaskan isi komponen yang ada pada sebuah sistem yang ingin dibuat. Berikut salah satu contoh pemodelan penggambaran sebuah Component Diagram.



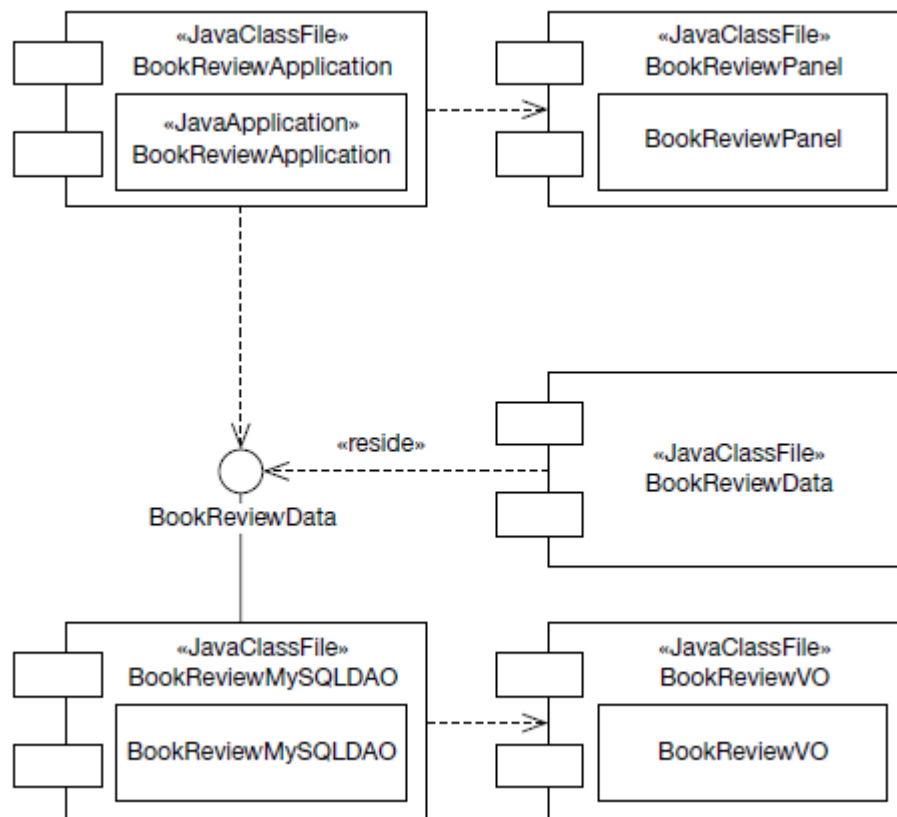
Gambar 39. Contoh pemodelan pada Component Diagram.

Setiap komponen model menunjukkan beberapa class dan interface bertugas sebagai komponen. Fungsi ini dapat menunjukkan penugasan tiap class komponen maupun secara fisik yang digambarkan dengan ikon, ataupun sebagai ketergantungan antara komponen dan class pada gambar diatas tertulis '`<<reside>>`'. Seperti contoh diatas komponen C bergantung dengan komponen D. Pada Component Diagram hanya ada *descriptor form*. Dikarenakan komponen adalah bagian fisik yang ada pada suatu sistem. Hanya komponen hanya dapat diunjukkan dengan sebuah *instance*/contoh yang tergabung kedalam wujud perangkat fisik.

a. Contoh pada sistem Java

Disesi kali ini akan mengilustrasikan UML component yang ada pada sistem Java. Akan ditunjukkan pemodelan simple yang ada pada aplikasi Java database. Aplikasi ini menggunakan *Data Access Objec* (DAO). Rencana dibalik pola ini DAO akan membuat sebuah kode akses secara spesifik ketika mengakses kedalam database. Aktifitas DAO terjadi dibelakang layar atau tidak berada dalam interface. Yang nantinya proses pada DAO akan menghasilkan sebuah data visual kedalam *interface*/layarmuka. Dengan membuat proses DAO didalam database, pengguna tidak akan kebingungan ketika mengakses suatu data yang ingin ditampilkan pada *interface*.

Di Java, file kelas biasanya berisi kode untuk sebuah java class dan ini membuat penggambaran hubungan antara class dan komponen menjadi sederhana. Berikut implementasi contoh Component Diagram pada sebuah aplikasi.



Gambar 40. Component Diagram pada aplikasi berbasis Java.

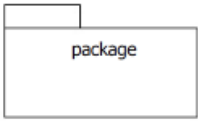
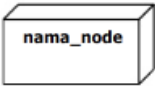


BookReviewApplication merupakan komponen yang berisi Class tunggal dan juga disebut *BookReviewApplication*. Dan kelas tersebut berada pada komponen *JavaApplication* atau terindikasi dengan program yang dapat dieksekusi pada sebuah program Java. Komponen ini menggunakan *BookReviewPanel* komponen yang mempunyai class yang sama, dan berada pada GUI (*Graphic User Interface*) pada sebuah aplikasi. Class tersebut memiliki ketergantungan antara dua komponen, daripada yang ditampilkan ditampilkan, yang dimana komponen terkait erat dan berada di *logical layer* yang sama.

Komponen *BookReviewApplication* menggunakan tampilan dari *BookReviewData* yang berada didalam komponen *BookReviewData*. Tampilan *BookReviewData* diimplementasikan oleh komponen

BookReviewMySQLDAO. Komponen ini adalah sebuah DAO yang menyediakan akses ke *book Review data* yang tersimpan didalam database berelasi MySQL. Fungsi DAO lainnya dapat disediakan untuk merelasikan database lainnya asalkan keseluruhannya harus mengimplementasikan kedalam tampilan *BookReviewData*. Semua informasi file dapat dipertukarkan selama aplikasi masih bersangkutan.

b. Deployment Diagram

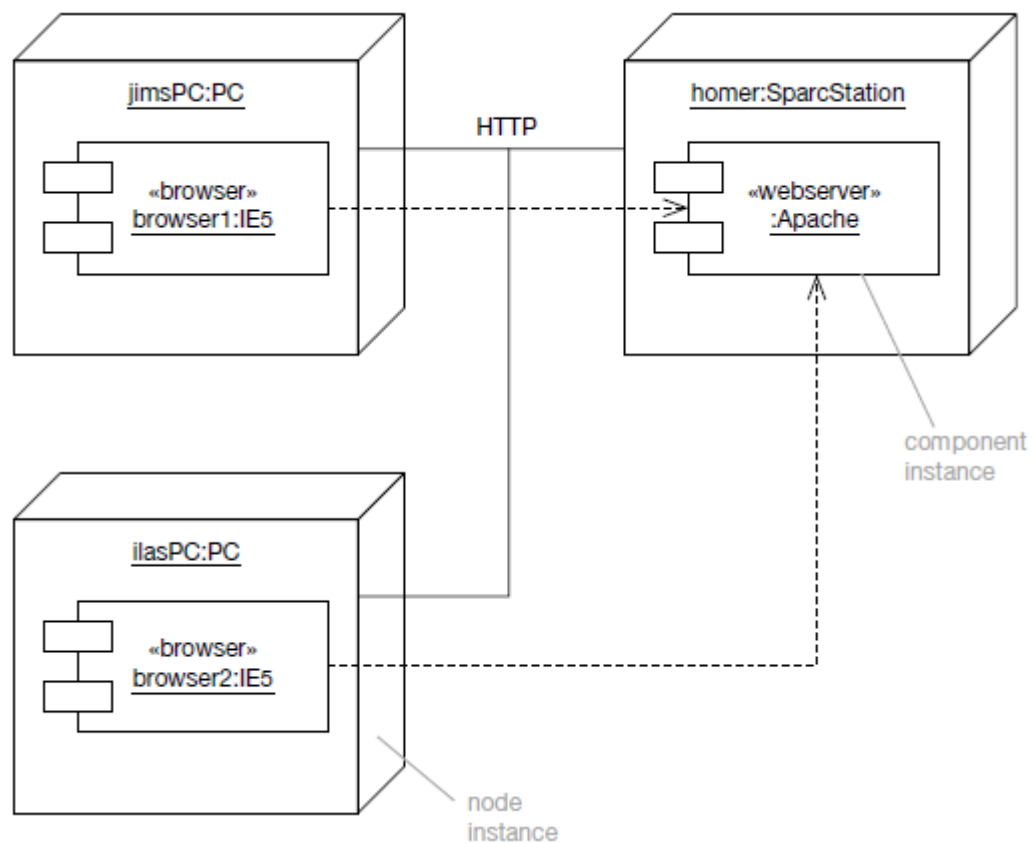
Deployment diagram adalah diagram uml yang dipakai untuk menjelaskan dan menggambarkan, menspesifikasikan serta menyimpan suatu process yang terjadi disebuah aplikasi berbasis OOP (*Object Oriented Programming*) yang akan dibangun. Dapat diartikan juga deployment diagram adalah sebuah pendeskripsian proses pada aplikasi yang sedang berlangsung terjadi serta menjelaskan bagaimana hubungan relasi di dalam sistem tersebut. Seperti yang diterangkan sebuah fungsi deployment diagram yaitu mendeskripsikan dan menggambarkan dan menspesifikasi proses yang terjadi pada sistem yang ingin dirancang. Salah satu contohnya ketika ingin menspesifikasikan sebuah situs web, maka yang perlu diperhatikan yaitu perangkat keras yang digunakan atau sebut saja dengan node pada kasus ini. Contohnya seperti Database server, Server Aplikasi, Web Server. Akan dijelaskan simbol-simbol yang ada dideployment diagram.

Simbol	Deskripsi
Package 	package merupakan sebuah bungkus dari satu atau lebih <i>node</i>
Node 	biasanya mengacu pada perangkat keras (<i>hardware</i>), perangkat lunak yang tidak dibuat sendiri (<i>software</i>), jika di dalam <i>node</i> disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / dependency 	Kebergantungan antar <i>node</i> , arah panah mengarah pada <i>node</i> yang dipakai
Link 	relasi antar <i>node</i>

Tabel 2. Simbol pada deployment Diagram

Didalam Package terdapat node-node yang biasanya mengacu pada perangkat keras, perangkat lunak. Komponen tersebut nantinya akan saling bergantung dengan node lain, serta dapat berelasi dengan node lain.

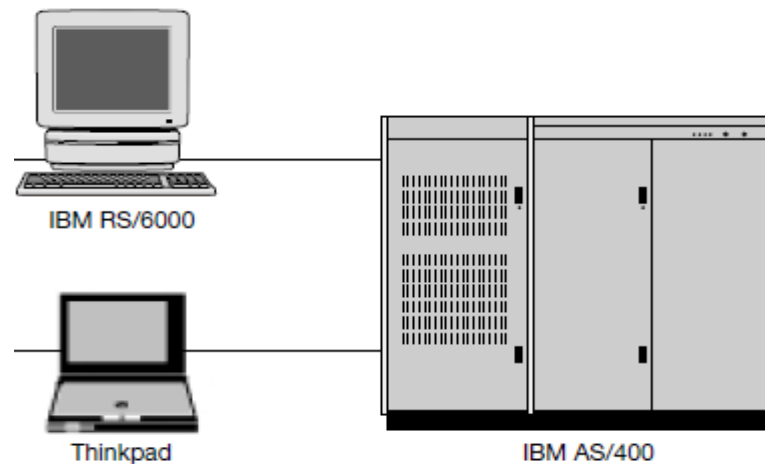
Akan dijelaskan juga contoh sederhana pengimplementasian sederhana deployment diagram yang ada pada gambar berikut.



Gambar 41. Contoh deployment diagram sederhana.

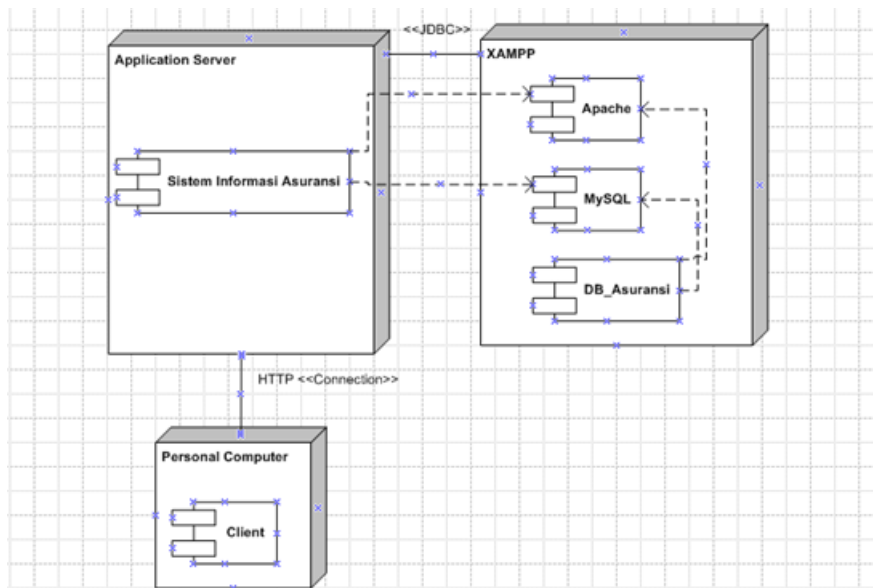
Contoh sederhana ketika seseorang ingin mengakses sebuah website sparcstation, 'jimsPC:PC' kita anggap sebuah node atau perannya disini sebagai perangkat keras, 'homer:SparcStation' sebagai alamat website yang ingin diakses oleh 'jimsPC' lewat browser, website 'homer:SparcStation' tersebut dihosting dengan super computer. Kemudian ada ilasPC:PC yang berperan sebagai admin yang mempunyai akses secara keseluruhan terhadap website homer:SparcStation dan dapat mengedit semua elemen yang ada pada website homer:SparcStation melalui device yang ia punya baik itu komputer ataupun laptop.

Jika digambarkan secara fisik maka akan terlihat seperti gambar dibawah berikut ini.



Gambar 42. Gambaran perangkat keras yang digunakan pada gambar sebelumnya

Selanjutnya akan ada studi kasus penggunaan Deployment diagram pada Sistem Informasi Asuransi.



Gambar 43. Deployment Diagram pada sistem asuransi

Bisa dilihat yang digambarkan pada Deployment diagram yaitu relasi koneksi hubungan antara, aplikasi user interface yang digunakan pengguna dengan server yang akan diakses. Kemudian dari aplikasi server, sistem akan mengambil informasi menuju database langsung

C. SOAL LATIHAN/TUGAS

1. Carilah Definisi dari Sequence Diagram, Collaboration Diagram, Component Diagram, Deployment Diagram selain yang disebutkan diatas!
2. Jelaskan perbedaan antara Sequence Diagram & Collaboration Diagram menurut kalian!
3. Jelaskan perbedaan antara Component Diagram & Deployment Diagram menurut kalian!
4. Buatlah Studi kasus contoh untuk pembuatan Sequence Diagram!
5. Buatlah Studi kasus contoh untuk pembuatan Collaboration Diagram!
6. Buatlah Studi kasus contoh untuk pembuatan Component Diagram!
7. Buatlah Studi kasus contoh untuk pembuatan Deployment Diagram!
8. Buatlah Relasi Deployment Diagram yang menggambarkan secara fisik!

D. REFERENSI

- John W. Satzinger, Robert B. Jackson, Stephen D.Burd (2016). *Systems Analysis and Design In A Changing World*. 7th edition. Boston: Cengage Learning.
- Jim Arlow, Ila Neustadt (2002). *UML And The Unified Process Practical Object-Oriented Analysis & Design*. Great Britain: Pantek Arts,Ltd, Maidstone, Kent

GLOSARIUM

Sequence merupakan sebuah antrian yang harus dilalui ketika sistem berjalan

Message merupakan sebuah pesan atau pendeskripsian hal-hal apa saja yang terjadi pada sebuah sequence diagram.

Participant adalah sebuah elemen pada sequence diagram, yang tiap elemennya mempunyai tugas masing-masing.

Source file adalah sebuah sumber data yang ada pada database atau penyimpanan data.

JavaBeans merupakan sebuah elemen suatu program yang menggunakan bahasa Java.

Java merupakan bahasa pemrograman dasar pada JDK (Java Development Kit)

Object oriented programming merupakan bahasa pemrograman yang berfokus pada orientasi serta objek.

Database Server merupakan server penyimpanan data yang dipakai untuk kebutuhan sebuah sistem ataupun website.

Web Server merupakan server yang berfungsi untuk menghosting suatu website agar dapat dipublikasikan kedalam jaringan internasional atau *Internet*.

Pc merupakan istilah lain dari personal computer, atau perangkat komputer milik pribadi.

Xampp adalah sebuah software yang dapat menghosting suatu database, ketika seseorang membuat aplikasi berbasis *CRUD* (*create, read, update, delete*).

MySql merupakan salah satu elemen aplikasi yang berada pada Xampp, berfungsi sebagai penyimpanan data.

PERTEMUAN 14

STUDI KASUS PERANCANGAN DIAGRAM UML PEMBUATAN SUATU SISTEM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan langsung Studi kasus perancangan diagram uml pada sebuah studi kasus, agar mahasiswa dapat mempelajari dan mendapatkan sebuah gambaran ketika ingin merancang suatu sistem.

B. URAIAN MATERI

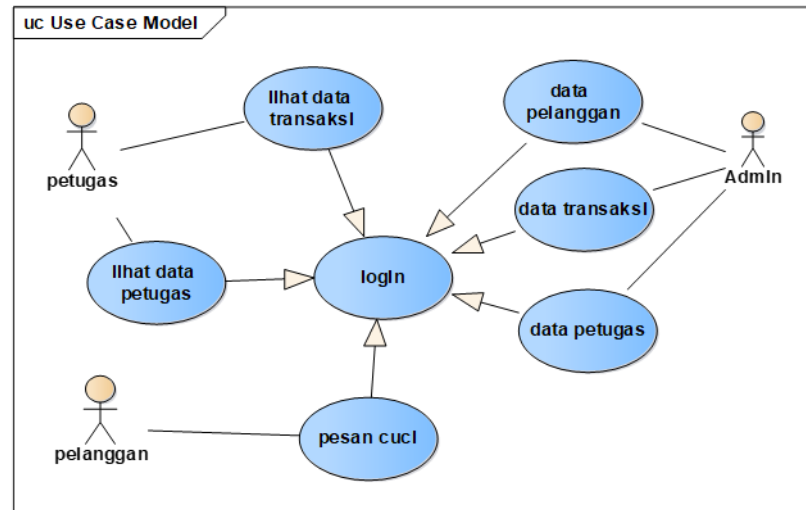
1. Perancangan Studi kasus Dilapangan

Kali ini akan mengambil studi kasus sistem aplikasi pelayanan pencucian kendaraan berbasis *Android*. Yang nantinya akan membantu proses bisnis yang berjalan pada usaha pencucian kendaraan. Memudahkan ketika menyimpan data, manajemen User, manajemen Karyawan, dan Laporan keuangan bulanan.

2. Perancangan Use Case Diagram

Use Case merupakan perancangan yang penting pada kasus ini, dikarenakan memberikan suatu gambaran hal-hal apa saja yang akan dibutuhkan pada aplikasi ketika aplikasi atau sistem akan dibuat, selain itu juga sebagai acuan desain ketika membuat aplikasi, lalu berfungsi juga membatasi beberapa hal-hal persyaratan yang dapat divalidasi ketika aplikasi dibuat suatu rancangan.

Berikut dibawah ini gambaran Use Case pada studi kasus pembuatan aplikasi pencucian kendaraan.



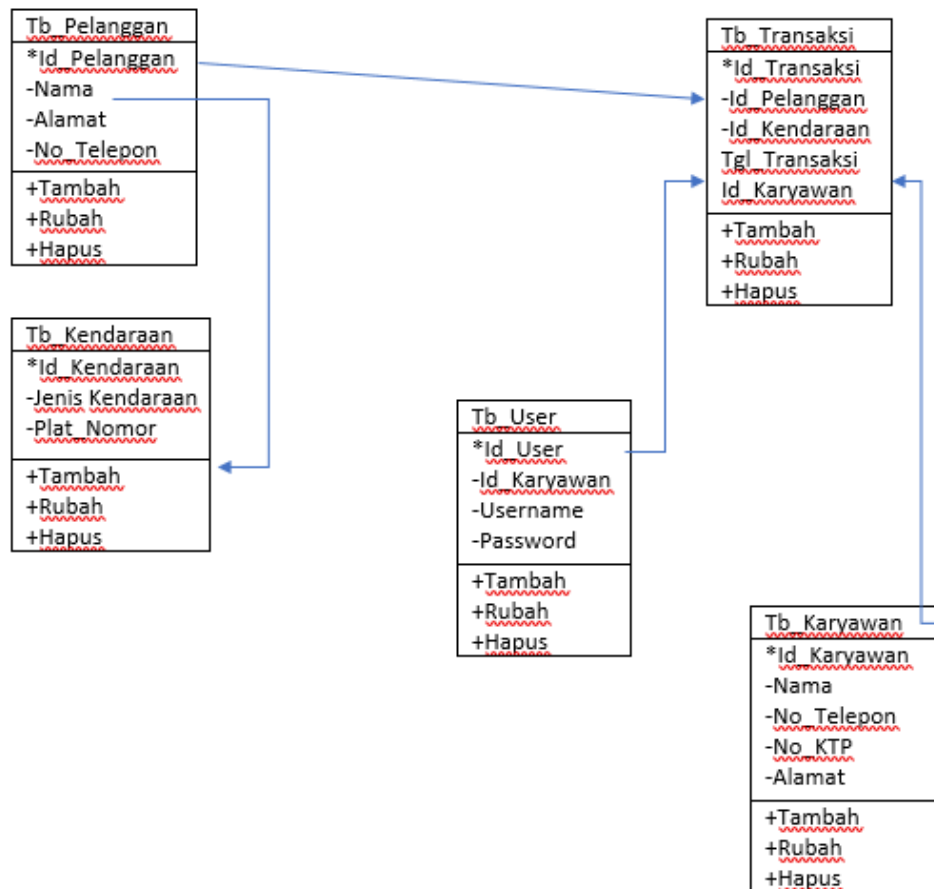
Gambar 44. Use Case pada sistem pelayanan pencucian kendaraan berbasis *Android*

Dengan adanya Use Case dapat mengidentifikasi aktor-aktor apa saja yang terlibat dalam suatu sistem yang berada pada ruang-lingkup sebuah sistem.

3. Perancangan Class Diagram

Class diagram yaitu sekumpulan obyek yang dimana setiap obyeknya memiliki atribut dan operasi. Class Diagram itu Sendiri suatu diagram yang dimana didalamnya terdiri dari objek-objek yang masing-masing mempunyai atribut serta operasi. Pada studi kasus ini akan memperjelas tiap-tiap atribut dan sistem operasi untuk mempermudah mengidentifikasi operasi apa saja yang dapat dijalankan pada tiap tiap atribut.

Berikut ini contoh Class Diagram pada aplikasi pelayanan pencucian kendaraan.



Gambar 45. Class Diagram Aplikasi pelayanan pencucian kendaraan

Tiap atribut memiliki operasi, dibawah ini akan dijelaskan rangkaian atribut yang ada pada sistem Pelayanan pencucian kendaraan.

Nama file : Tb_Pelanggan

Isi : Data-Data Pelanggan

Primary Key : Id_Pelanggan

Struktur File :

No	Nama File	Type	Ukuran	Keterangan
1	Id_Pelanggan	Varchar	60	Id Pelanggan
2	Nama	Varchar	30	Nama Pelanggan
3	Alamat	Text		Alamat Pelanggan
4	No_Telepon	Varchar	25	Nomor Telepon

Tabel 3. Struktur Tabel Tb_Pelanggan

Nama File : Tb_Kendaraan

Isi : Data-data Kendaraan

Primary Key : Id_Kendaraan

Struktur File :

No	Nama File	Type	Ukuran	Keterangan
1	Id_Kendaraan	Varchar	60	Id Kendaraan
2	Jenis_Kendaraan	Varchar	30	Jenis Kendaraan
3	Plat_Nomor	Varchar	15	Plat Nomor Kendaraan

Tabel 4. Struktur Tabel Tb_Kendaraan

Nama File : Tb_Transaksi

Isi : Riwayat Transaksi Pembayaran

Primary Key : Id_Transaksi

Struktur File :

No	Nama File	Type	Ukuran	Keterangan
1	Id_Transaksi	Varchar	60	Id transaksi
2	Id_Pelanggan	Varchar	60	Id pelanggan
3	Id_Kendaraan	Varchar	60	Id Kendaraan
4	Tgl_Transaksi	Date		Tanggal Transaksi

5	Id_Karyawan	Varchar	60	Id Karyawan
---	-------------	---------	----	-------------

Tabel 5. Struktur Tabel Tb_Transaksi

Nama File : Tb_User

Isi : Data para Pengguna

Primary Key : Id_User

Struktur File :

No	Nama File	Type	Ukuran	Keterangan
1	Id_User	Varchar	60	Id pengguna
2	Id_Karyawan	Varchar	60	Id Karyawan
3	Username	Varchar	30	Nama User
4	Password	Varchar	25	Kata Kunci

Tabel 6. Struktur Tabel Tb_User

Nama File : Tb_Karyawan

Isi : Data-data Karyawan

Primary Key : Id_Karyawan

Struktur File :

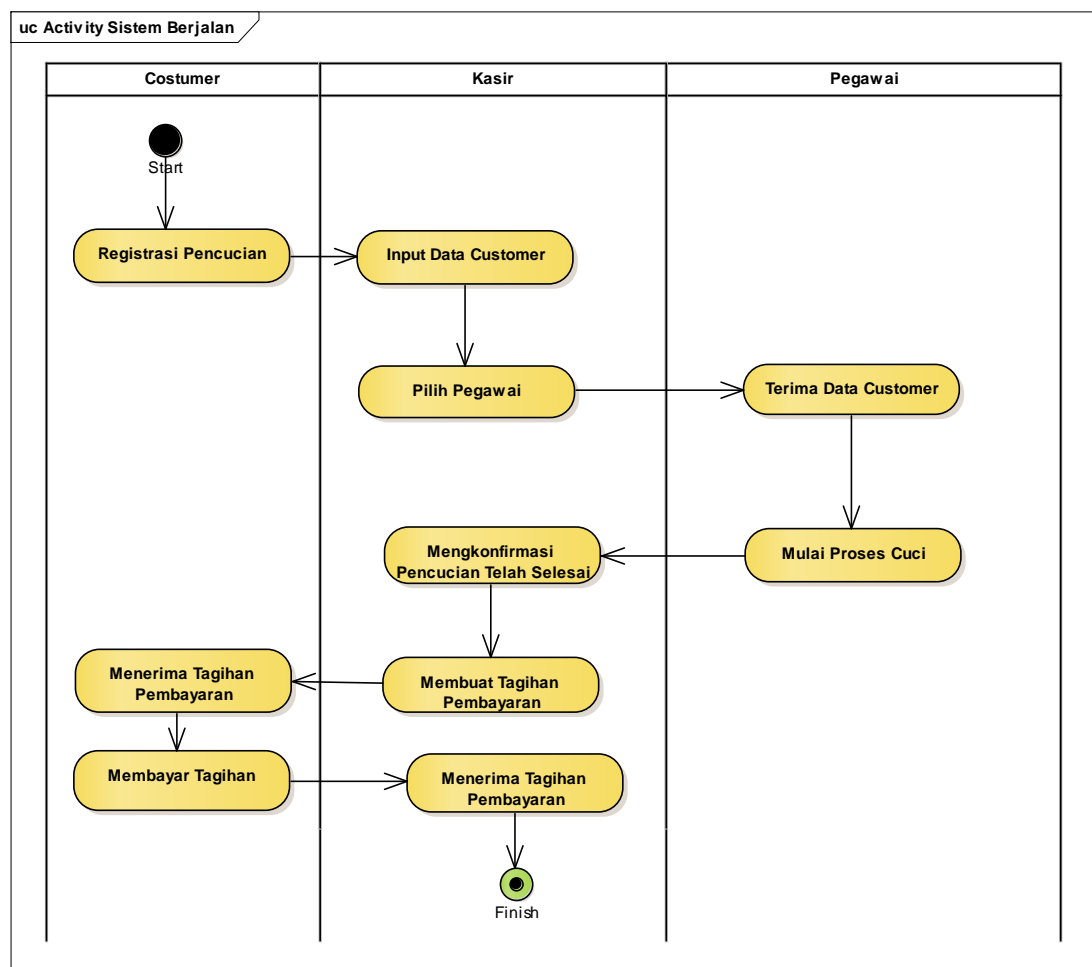
No	Nama File	Type	Ukuran	Keterangan
1	Id_Karyawan	Varchar	60	Id Karyawan
2	Nama	Varchar	30	Nama Karyawan
3	No_Telepon	Varchar	25	Nomor Telepon
4	No_Ktp	Varchar	30	Nomor kartu keterangan penduduk
5	Alamat	Text		Alamat Karyawan

Tabel 7. Struktur table Tb_Karyawan

4. Activity Diagram

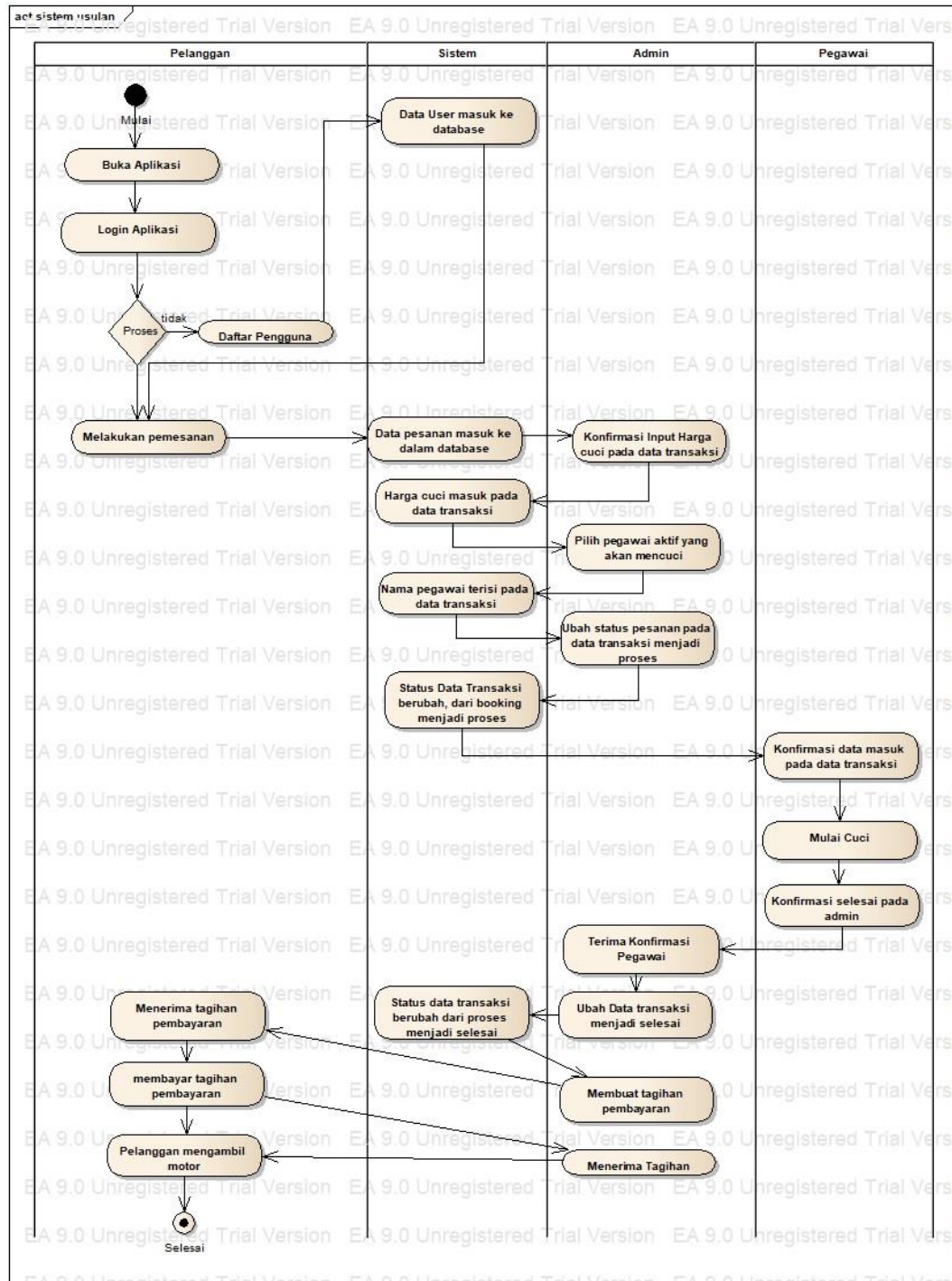
Activity Diagram merupakan bagian terpenting pada UML, yang menggambarkan aspek dinamis dari sistem dan menggambarkan gambaran suatu aktifitas pada sistem yang jelas membuat penggunanya lebih memahami alur fungsi pada suatu rancangan aplikasi yang akan dirancang. Oleh karena itu distudi kasus kali ini diperlukan sebuah Activity Diagram untuk menggambarkan proses berjalannya pada suatu sistem secara jelas.

Berikut penggambaran Activity Diagram sistem pelayanan pencucian kendaraan sebelum sistem diimplementasikan dengan sistem yang akan diajukan nantinya.



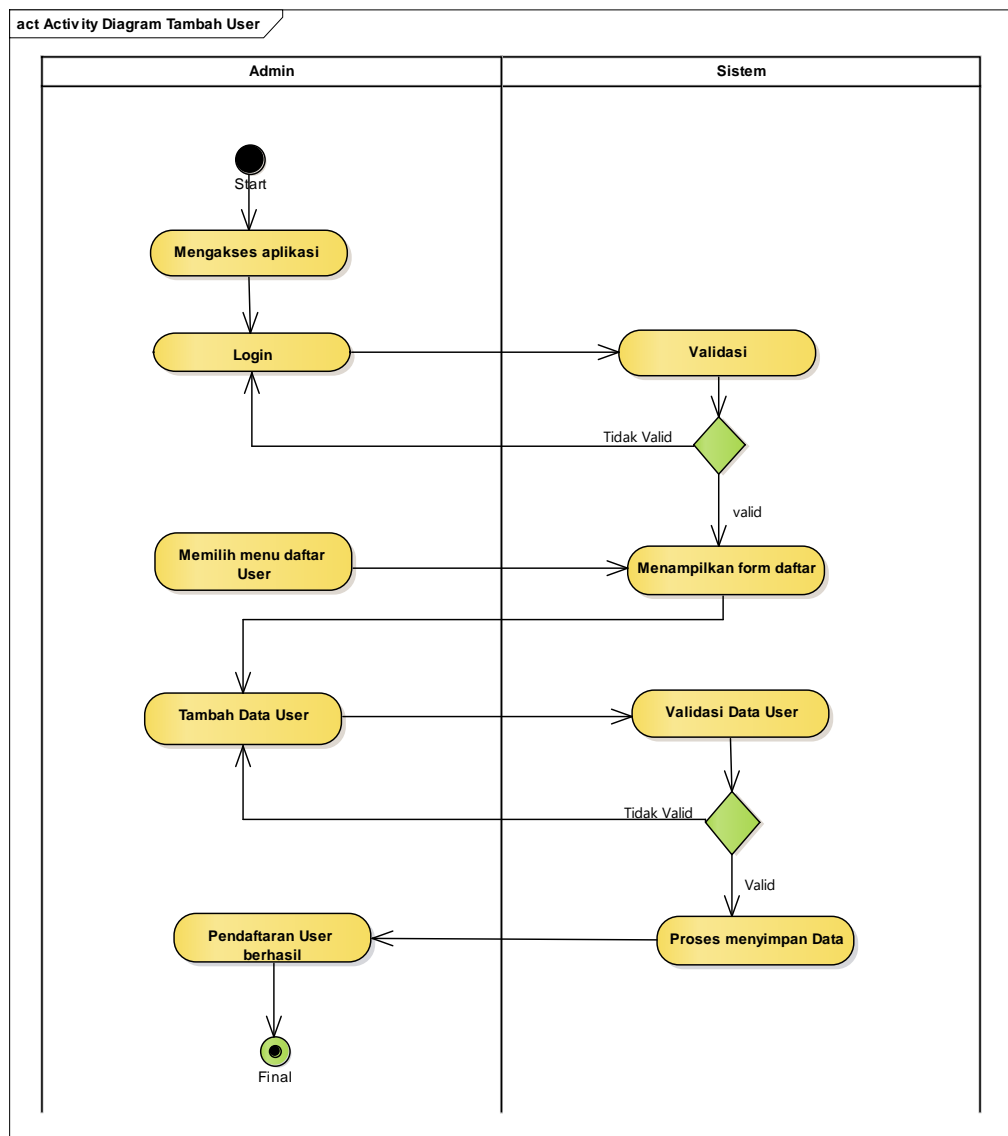
Gambar 46. Activity Diagram sistem berjalan pada sistem pelayanan pencucian kendaraan

Selanjutnya sistem usulan yang akan diajukan untuk mengimplementasikan aplikasi yang akan dibuat.



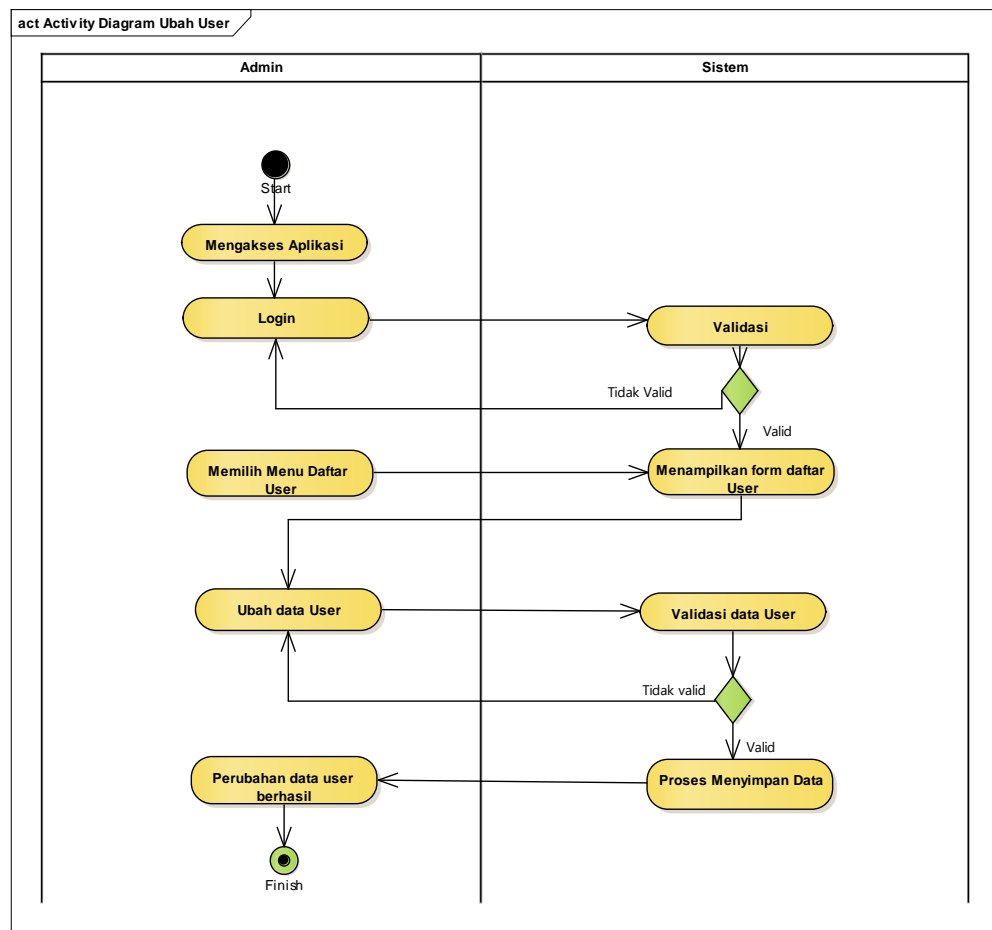
Gambar 47. Activity Diagram sistem usulan pada sistem pencucian kendaraan

Selanjutnya activity diagram tambah user pada fungsi registrasi user.



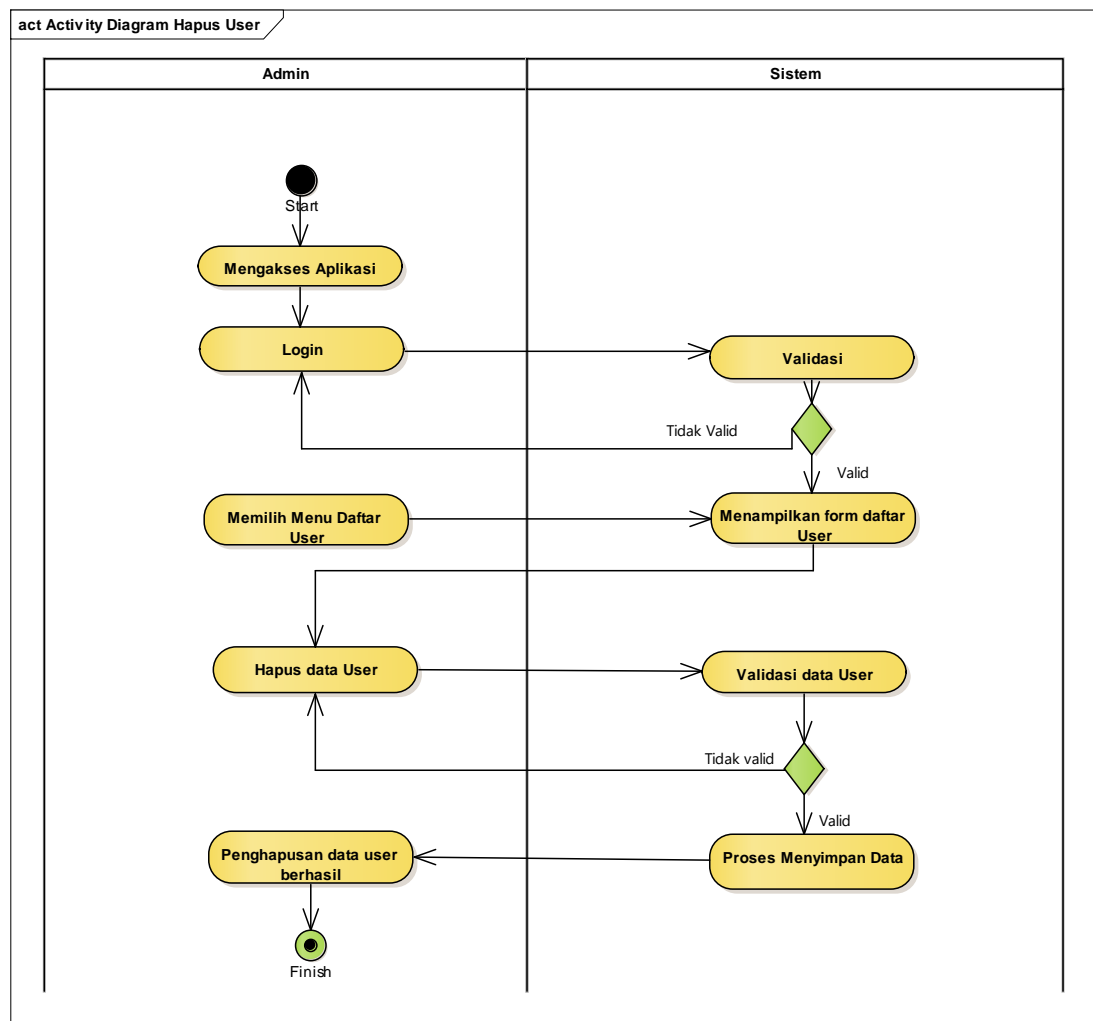
Gambar 48. Activity Diagram registrasi user

Lalu Activity Diagram untuk ubah user.



Gambar 49. Activity Diagram Ubah User

Lalu Activity Diagram Hapus user



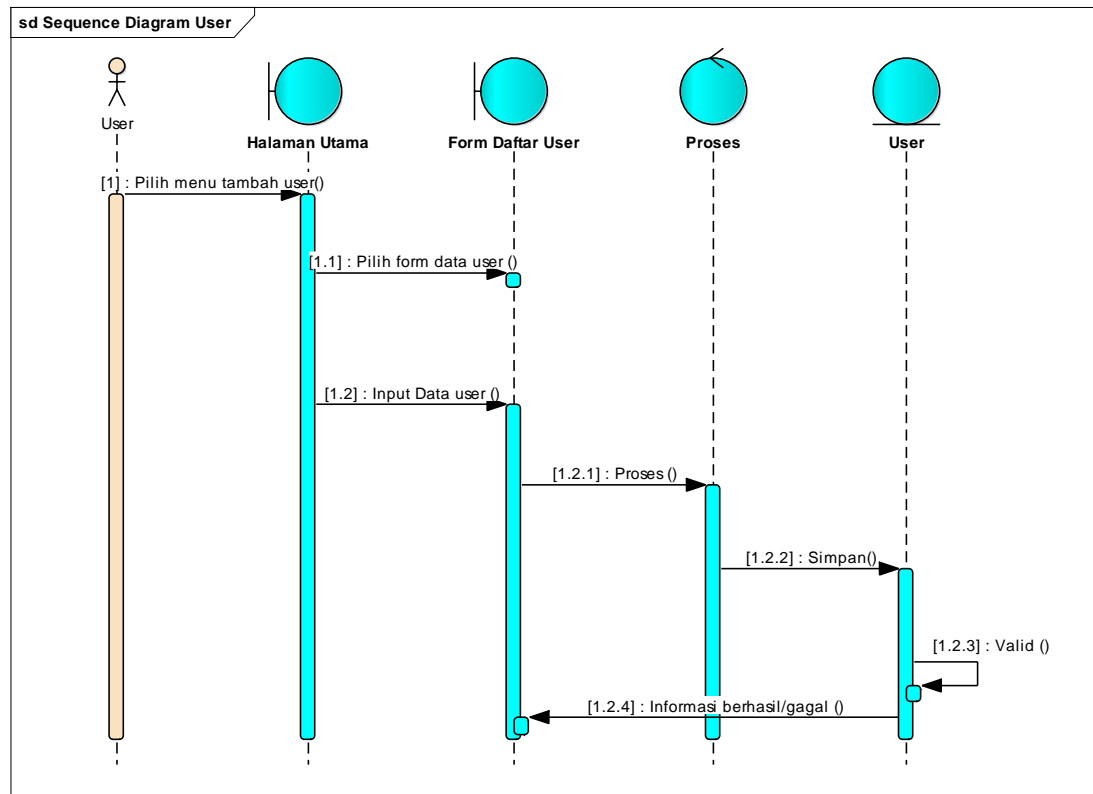
Gambar 50. Activity Diagram Hapus User

5. Sequence diagram

Sequence Diagram diperlukan guna mendeskripsikan tingkah laku yang berada di sebuah sistem. Diagram ini menjelaskan sejumlah objek dan pesan yang diletakkan diantara beberapa objek yang berada di suatu use case. Elemen inti sequence diagram biasa tertulis dengan berwarna kotak atau lingkaran bernama. Tiap pesan atau *Message* terwakili dengan garis tanda panah dan waktu yang ditunjukkan dengan proses vertical. Obtek diletakkan dekat bagian atas diagram mengurut dari kiri kerah kanan. Diatur dalam urutan yang menyederhanakan diagram. Dengan adanya sequence diagram dapat mengidentifikasi alur berjalannya suatu proses dari awal hingga akhir. Berikut

gambaran sistem diagram yang ada pada sistem pelayanan pencucian kendaraan.

Sequence Diagram Pendaftaran User

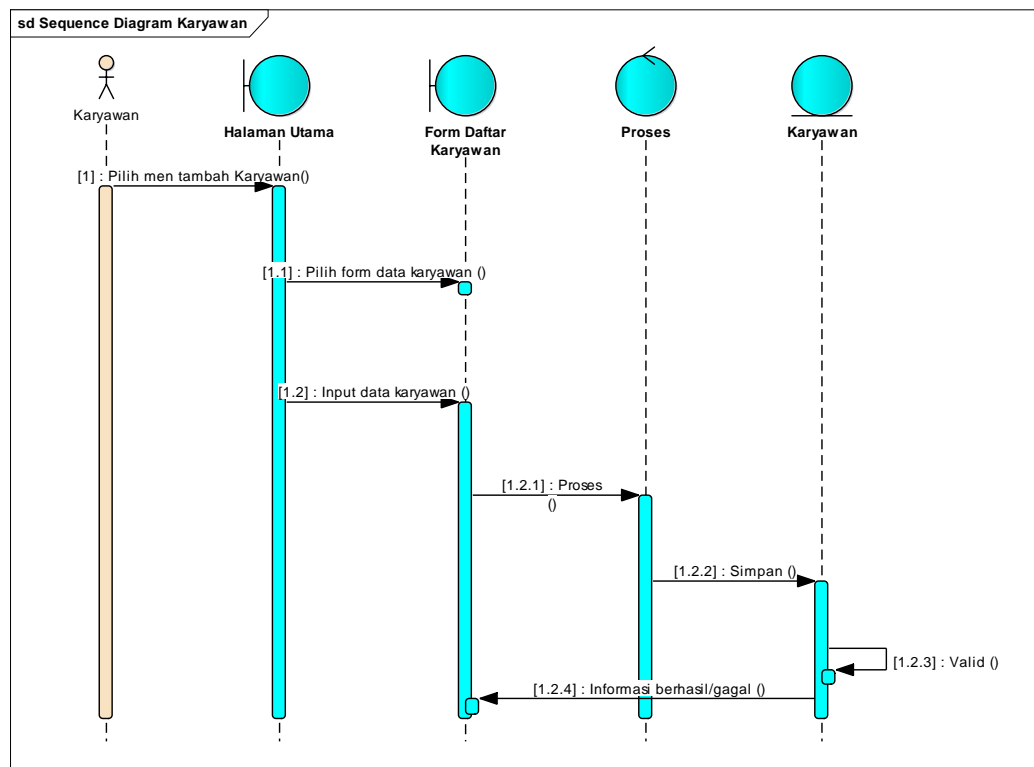


Gambar 51. Sequence Diagram Pendaftaran user.

Diagram Diatas mengenai proses Kelola pendaftaran *User*. Proses untuk gambar diatas yaitu :

Admin memilih menu daftar, *Admin* dapat melakukan pendaftaran dengan mengisi data di *Form* pendaftaran, Sistem melakukan proses penyimpanan data ke database *User*.

Sequence Diagram dari pendaftaran karyawan.

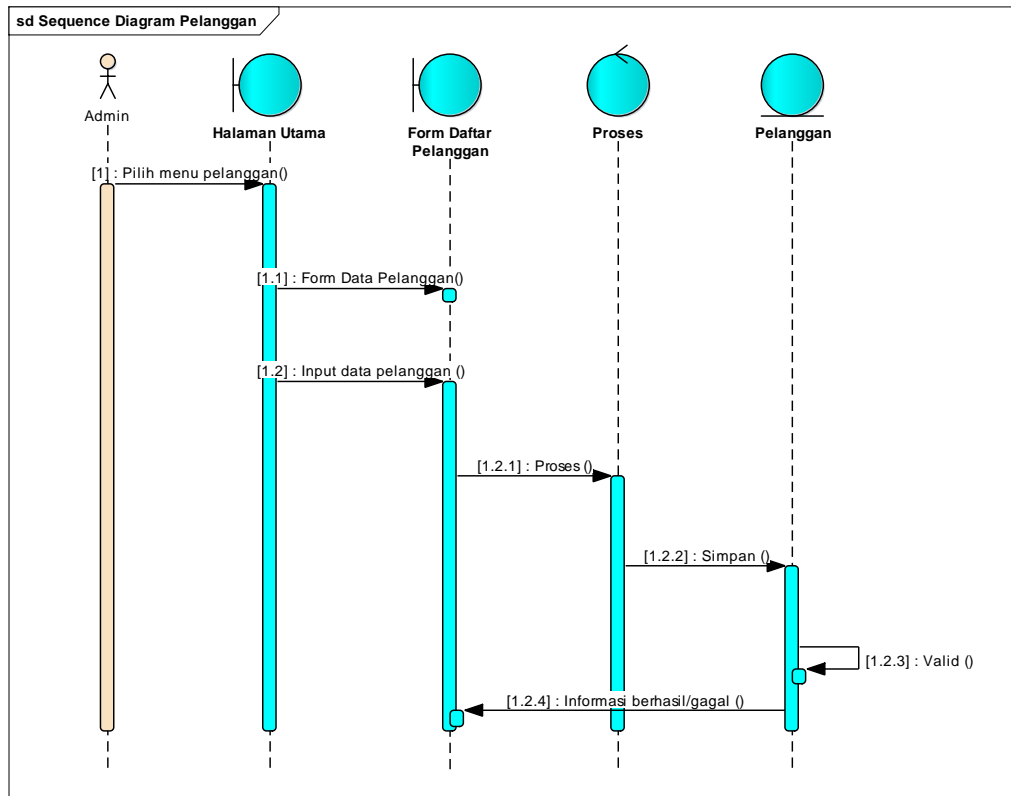


Gambar 52. Sequence diagram pendaftaran Karyawan

Diagram diatas mengenai proses Kelola pendaftaran Karyawan. Proses untuk gambar diatas yaitu :

Admin memilih menu daftar, *Admin* dapat melakukan dengan mengisi data pada *Form* daftar, kemudian Sistem akan melakukan proses penyimpanan data ke database karyawan.

Sequence Diagram pendaftaran pelanggan.

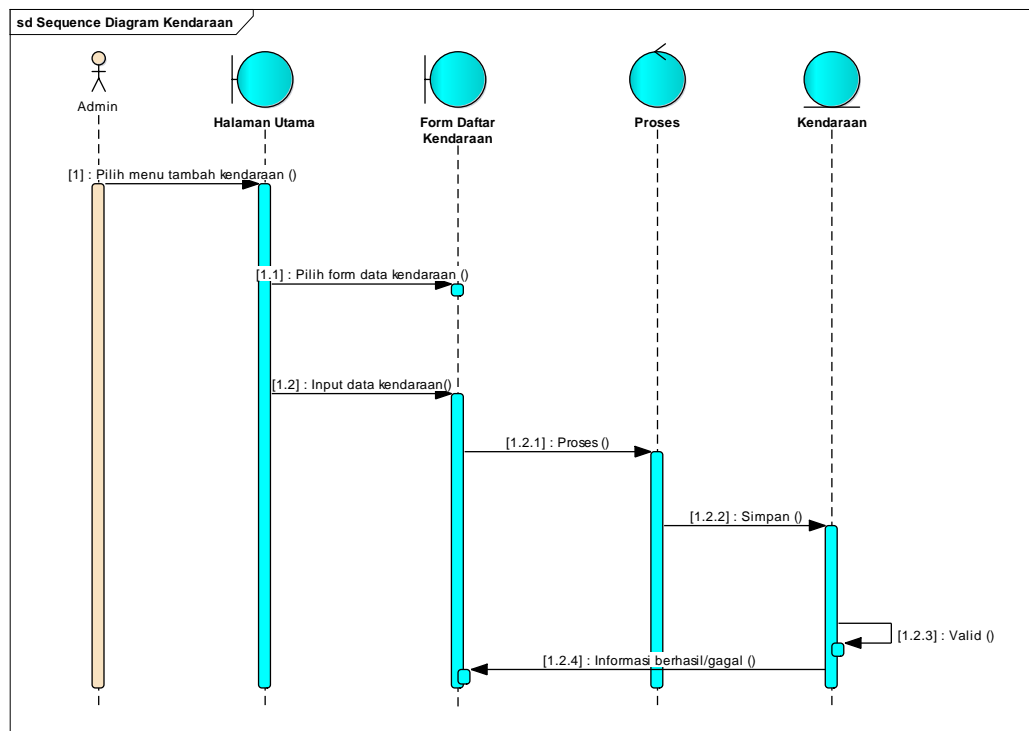


Gambar 53. Sequence Diagram pendaftaran pelanggan

Diagram diatas mengenai proses Kelola pendaftara pelanggan. Proses untuk gambar diatas yaitu :

Admin memilih menu daftar, *Admin* dapat melakukan pendaftaran dengan mengisi data pada *Form* daftar, Kemudian sistem akan melakukan proses penyimpanan data ke database pelanggan.

Sequence diagram pendaftaran kendaraan.



Gambar 54. Sequence diagram pendaftaran Kendaraan.

Itulah beberapa elemen-elemen yang diperlukan untuk studi kasus perancangan sistem pelayanan pencucian kendaraan berbasis *Android*. Perlu diketahui bahwa suatu perancangan tidaklah harus memakai keseluruhan *UML* diagram tergantung pada sistem yang ingin dirancang.

C. SOAL LATIHAN/TUGAS

Buatlah perancangan *UML* melalui studi kasus yang ada disekitar lingkungan kalian !
sss

D. REFERENSI

John W. Satzinger, Robert B. Jackson, Stephen D.Burd (2016). *Systems Analysis and Design In A Changing World*. 7th edition. Boston: Cengage Learning.

Jim Arlow, Ila Neustadt (2002). *UML And The Unified Process Practical Object-Oriented Analysis & Design*. Great Britain: Pantek Arts,Ltd, Maidstone, Kent

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc

GLOSARIUM

Android merupakan sistem operasi yang terinstalasi perangkat mobile atau telepon genggam pintar android.

Record adalah kumpulan dari elemen data yang berhubungan dalam kesatuan pada basis data.

DataBase merupakan media yang dapat menyimpan suatu file dan terhubung oleh server.

Admin merupakan pengguna yang memiliki kontrol penuh pada suatu sistem atau aplikasi.

User seorang pengguna aplikasi yang mempunyai kontrol khusus terhadap suatu aplikasi.

Primary Key adalah suatu nilai dalam basis data yang bersifat unik dan dapat difungsikan untuk membedakan antara sebuah record dengan record lainnya.

RENCANA PEMBELAJARAN SEMESTER (RPS)

Program Studi : Teknik Informatika

Mata Kuliah/Kode : Analisa dan Perancangan Sistem /TPL0282

Prasyarat : -

SKS : 2 SKS

Deskripsi Mata Kuliah : Matakuliah ini merupakan mata kuliah Wajib Program Studi teknik Informatika S-1 yang membahas pengertian sistem, pengembangan sistem, analisis sistem, data flow diagram, flowchart, metodologi berorientasi obyek, diagram-diagram UML dan melakukan perancangan sistem dengan model UML.

Capaian Pembelajaran : Setelah menyelesaikan mata kuliah ini, mahasiswa mampu merancang, mendisain dan menganalisis sistem dengan Metodologi Berorientasi Obyek serta diharapkan dapat mendisain sistem dengan diagram-diagram UML sesuai dengan kebutuhan sistem.

Penyusun : 1. Irpan Kusyadi (Ketua)
2. Maulana Ardhiansyah (Anggota 1)
3. Hidayatullah Al Islami (Anggota 2)

PERTEMUAN KE-	KEMAMPUAN AKHIR YANG DIHARAPKAN	BAHAN KAJIAN (MATERI AJAR)	METODE PEMBELAJARAN	PENGALAMAN BELAJAR MAHASISWA	KRITERIA PENILAIAN	BOBOT NILAI
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	Mahasiswa mampu mendeskripsikan klasifikasi dan metodologi pengembangan sistem	1.1 Pengertian Sistem 1.2 Karakteristik Sistem 1.3 Klasifikasi Sistem 1.4 Pendekatan dan Metodologi Pengembangan Sistem	Inquiry Diskusi Presentasi	Mengerjakan tugas dengan mencari sumber melalui Internet	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ kebenaran substansi ▪ tanggung jawab ▪ disiplin 	7
2	Mehasiswa mampu memahami System Development Life Cycle (SDLC)	2.1 Pengertian SDLC 2.2 Sejarah Pengembangan SDLC 2.3 Tahapan SDLC 2.4 Alat Pengembangan Sistem	Inquiry Diskusi Presentasi	Mengerjakan tugas dengan mencari sumber melalui Internet, melakukan analisa dan mengambil kesimpulan	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ kebenaran substansi ▪ tanggung jawab ▪ disiplin 	7

3	Mahasiswa mampu mendeskripsikan perencanaan sistem	3.1 Pengertian Perencanaan 3.2 Perlunya Perencanaan 3.3 Proses Perencanaan Sistem	Inquiry Survey Diskusi Presentasi	Mengerjakan tugas dengan mencari sumber melalui Internet, melakukan analisa dan mengambil kesimpulan	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
4	Mahasiswa mampu mendeskripsikan analisa sistem	4.1 Pengertian Analisa Sistem 4.2 Proses Analisa Sistem	Inquiry Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
5	Mahasiswa mampu memahami proses modelling dengan menggunakan DFD	5.1 Pengertian proses Modelling 5.2 Data Flow Diagram 5.3 Komponen Data Flow Diagram 5.4 Bentuk Data Flow	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7

		Diagram				
6	Mahasiswa mampu memahami sistem Flowchart	6.1 Pengertian sistem Flowchart 6.2 Simbol-simbol Flowchart 6.3 Jenis-jenis Flowchart	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
7	Mahasiswa mampu memahami konsep pendekatan perancangan berorientasi obyek	7.1 Konsep Perancangan berorientasi obyek 7.1 Konsep Perancangan berorientasi obyek 7.1 Konsep Perancangan berorientasi obyek 7.1 Konsep Perancangan berorientasi obyek 7.2 Konteks sistem dan model penggunaan 7.2 Konteks sistem dan model penggunaan 7.2 Konteks sistem dan model penggunaan	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7

		7.2 Konteks sistem dan model penggunaan 7.2 Konteks sistem dan model penggunaan 7.3 Perancangan arsitektural				
8	Mahasiswa mampu memahami konsep pendekatan perancangan berorientasi obyek	8.1 Identifikasi obyek 8.1 Identifikasi obyek 8.1 Identifikasi obyek 8.1 Identifikasi obyek 8.2 Model disain 8.3 Spesifikasi interface obyek	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
9	Mahasiswa mampu melakukan perancangan sistem berorientasi obyek menggunakan alat bantu UML (Unified Modelling Language)	9.1 Pengenalan UML 9.1 Pengenalan UML 9.1 Pengenalan UML 9.2 Sejarah Singkat UML	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7

10	Mahasiswa mampu melakukan perancangan sistem berorientasi obyek menggunakan alat bantu UML (Unified Modelling Language)	10.1 Bagian – bagian UML 10.1 Bagian – bagian UML 10.1 Bagian – bagian UML 10.2 Langkah – langkah pembuatan UML	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas, melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
11	Mahasiswa dapat membuat diagram-diagram UML sesuai dengan kebutuhannya	11.1 Use Case Diagram 11.1 Use Case Diagram 11.1 Use Case Diagram 11.2 Class Diagram 11.3 Object Diagram	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas studi kasus , melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
12	Mahasiswa dapat membuat diagram-diagram UML sesuai dengan kebutuhannya	12.1 Statechart Diagram 12.2 Activity Diagram	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas studi kasus , melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7

13	Mahasiswa dapat membuat diagram-diagram UML sesuai dengan kebutuhannya	13.1 Sequence Diagram 13.1 Sequence Diagram 13.1 Sequence Diagram 13.2 Collaboration Diagram 13.3 Component Diagram 13.4 Deployment Diagram	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas studi kasus , melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7
14	Mahasiswa mampu memahami konsep perancangan sistem dengan model UML dari sistem yang telah dirancang dan Mahasiswa mampu melakukan analisa dan mendisain sistem menggunakan model UML	Merancang sistem pada sebuah instansi dengan menganalisa dan mendisain sistem tersebut menggunakan diagram UML Sesuai dengan dibutuhkan	Inquiry Praktik Diskusi Presentasi	Mengerjakan tugas studi kasus , melakukan diskusi dan presentasi	<ul style="list-style-type: none"> ▪ Kreatifitas ▪ Kebenaran substansi ▪ tanggung jawab ▪ Disiplin ▪ Kerjasama 	7

Referensi/Sumber :

Dr. Jawahar. *Overview of System Analysis & Design*. Diakses dari: <http://www.ddegjust.ac.in/studymaterial/pgdca/ms-04.pdf>

Charles S. Wasson (2006). *System Analysis, Design, and Development: Concepts, Principles, and Practices*. New Jersey: John Wiley & Sons, Inc.

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Wikipedia (2020). *Software Development Process*. Diakses dari :
[https://en.wikipedia.org/wiki/Software_development_process#:~:text=The%20software%20development%20methodology%20\(also,framework%20for%20building%20information%20systems](https://en.wikipedia.org/wiki/Software_development_process#:~:text=The%20software%20development%20methodology%20(also,framework%20for%20building%20information%20systems).

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

Charles S. Wasson (2006). *System Analysis, Design, and Development: Concepts, Principles, and Practices*. New Jersey: John Wiley & Sons, Inc.

Jeffrey L. Whitten, Lonnie D. Bentley (2007). *Systems Analysis and Design Methods*. New York: The McGraw-Hill Companies, inc.

Shantanu Choudhary (2018). *Evolution of System Development Life Cycle (SDLC)*. Diakses dari:
<https://www.linkedin.com/pulse/evolution-system-development-life-cycle-sdlc-shantanu-choudhary>.

Langer, Arthur M. (2008). *Analysis and Design of Information Systems 3rd edition*. Switzerland: Springer.

Burch, J.G., System, Analysis, Design, and Implementation, Boyd & Fraser Publishing Company, 1992.

John G. Burch, Jr, Felix R. Strater, Gary Grudnitski, Information Systems : Theory and Practice, Second Edition, John Wiley & Sons, 1979.

- Meilir Page-Jones, The Practical Guide to Structured Systems Design, Second Edition, Yourdon Press, Prentice Hall, 1988.
- I.T. Hawryszkiewicz, Introduction Systems Analysis and Design, Second Edition, Prentice Hall, 1991
- Raymond McLeod, Jr, Management Information System : A Study of Computer-Based Information Systems, Sixth Edition, Prentice Hall, 1979
- A. Ziya Aktas, Structured Analysis & Design of Information Systems, NJ: Prentice Hall, 1987, hal. 65
- Dennis, Alan, Wixom, Barbara Haley, Roth, Roberta M. (2013). System Analysis and Design 5th edition. New Jersey: John Willey & Sons, Inc.
- IBM. (1969). Flowcharting techniques. (C20-8152-1 ed.). New York: IBM, Technical Publications Department.
- Tague, N. R. (2005). The quality toolbox. (2th ed.). Milwaukee, Wisconsin: ASQ Quality Press. Available from <http://asq.org/quality-press/displayitem/index.html?item=H1224>
- Yourdon Edward, Modern Structur Analisis, Prentice – Hall, Inc, 1989.
- Deutsches Institut für Normung. (September 1966). Sinnbilder für datenfluß- und programmablaufpläne. Deutsche Industrienorm DIN 66001. Tiergarten, Berlin: DIN.
- Rumbaugh James, et all, 1999, “The Unified Modeling Language Reference Manual”.
- Hoffer, A.Jeffrey and George, F, Joey, Modern System Analysis and Design, Prentice Hall-Inc, 2002
- McGraw Hill., UML, 1999
- Edward V. Berard, Origins Objects Oriented Technolog
- Grady Booch, 1991, Object-Oriented Analysis and Design with Application, Benjamin/Cummings.
- Dennis, Alan., Barbara Halley Wixom and Roberta M. Roth. 2012. System Analysis and Design 5 th Edition. John Willey and Sons, Inc. New Jersey

- Satzinger, John., Robert Jackson and Stephen Burd. 2010. *System Analysis and Design in Changing World* 5 th Edition. Cengage Learning. Boston.
- Practical UML A Hands-On Introduction for Developers,
Sri Dharwiyanti, Pengantar unified modeling language (UML),2003 IlmuKomputer.com
[http://www.togethersoft.com/services/practical_guides/umlonlinecourse/index.html]
- Unified Modeling Language Specification, Object Management Group, www.omg.org, 1999.
- Dpunkt.Verlag, Heidelberg, Germany, *An Introduction to Object-Oriented Modeling UML @ Classroom*, 2012.
- Introduction to OMG UML [http://www.omg.org/gettingstarted/what_is_uml.htm]
- UML Tutorial [http://www.sparxsystems.com.au/UML_Tutorial.htm]
- <http://share.its.ac.id/blog/index.php?entryid=689>
- Catur <https://garudacyber.co.id/artikel/1471-pengertian-uml-dan-komponen-uml>
- <http://www.materikuliahif-unpas.com/2018/07/sequence-diagram.html>
- Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc
- Jim Arlow, Ila Neustadt (2002). *UML And The Unified Process Practical Object-Oriented Analysis & Design*. Great Britain: Pantek Arts,Ltd, Maidstone, Kent
- John W. Satzinger, Robert B. Jackson, Stephen D.Burd (2016). *Systems Analysis and Design In A Changing World*. 7th edition. Boston: Cengage Learning.

Universitas Pamulang

Teknik Informatika

Tangerang Selatan, 20 Juli 2021

Ketua Program Studi

Ketua Tim Penyusun

Teknik Informatika

Achmad Udin Zailani, S.Kom., M.Kom.

Irpan Kusyadi, S. Kom., M. Kom.

NIDN : 0429058303

NIDN. 0411109001

