PDF-XChange
Click to buy NOW!
www.docu-track.com

Modul Kuliah

Analisis dan Desain Sistem Informasi

Daftar Isi

Daftar Gambar

Daftar Tabel

1 Pendahuluan

Overview

Bab ini merupakan pendahuluan sebelum menjelaskan inti materi buku ini terkait dengan analisis dan desain sistem informasi. Bab pendahuluan berisi mengenai definisi sistem informasi, sejarah perkembangan sistem informasi, tahapan pengembangan sistem informasi, siapa saja yang terlibat dalam pengembangan sistem informasi, serta penjadwalan pengembangan sistem informasi.

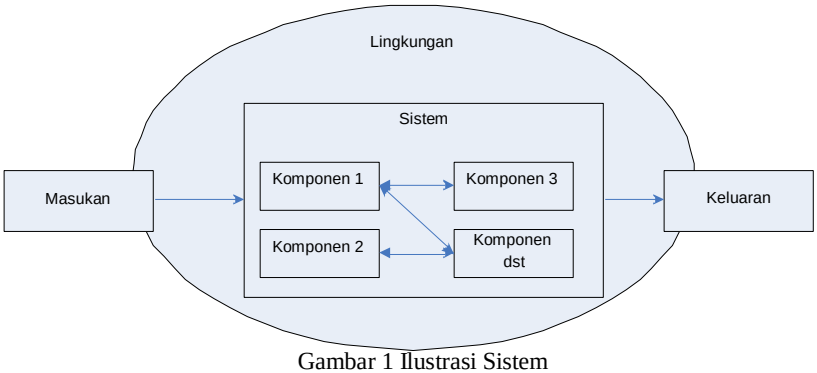
Tujuan

1. Mahasiswa mengetahui definisi dan sejarah perkembangan sistem informasi.
2. Mahasiswa mempunyai gambaran umum mengenai tahap-tahap yang dilalui dalam pengembangan sistem informasi.
3. Mahasiswa mempunyai gambaran umum mengenai analisis dan desain sistem informasi.

1.1 *Definisi Sistem Informasi*

Sistem ialah interaksi dari elemen-elemen yang saling berkaitan bekerja sama untuk mencapai tujuan. Elemen-elemen tersebut ialah elemen sistem konvensional (data, manusia dan prosedur) dan elemen sistem modern (data, manusia, prosedur, *hardware* dan *software*).

Ilustrasi mengenai sebuah sistem dapat dilihat pada gambar I. Sebuah sistem menerima masukan, memrosesnya, dan kemudian menghasilkan suatu keluaran. Sistem tersebut mampu bekerja karena komponen-komponen di dalamnya saling berinteraksi untuk menghasilkan keluaran. Dalam melakukan prosesnya, kinerja sistem sangat dipengaruhi oleh kondisi lingkungan di sekitarnya.



Gambar 1 Ilustrasi Sistem

Informasi ialah hasil pengolahan data yang berguna bagi penerimanya.

Sistem informasi ialah interaksi antara data, manusia dan prosedur (yang didukung oleh *hardware* dan *software*) untuk memberikan suatu penyelesaian berupa informasi yang dapat dipakai untuk mengambil suatu tindakan keputusan selanjutnya baik untuk jangka pendek, menengah atau panjang dalam sebuah organisasi. Dengan kata lain, sistem informasi juga adalah suatu kumpulan dari komponen-komponen yang saling berinteraksi untuk mengelola informasi pada suatu organisasi untuk mendukung kegiatan bisnis organisasi.

Pada awalnya sistem informasi tidak harus dikaitkan dengan teknologi informasi, namun seiring perkembangan jaman, saat ini suatu sistem informasi tidak dapat lepas dari penggunaan teknologi informasi.

Penggunaan teknologi informasi pada suatu sistem informasi mulai berkembang sekitar tahun 1960an. Pada periode tersebut, sistem informasi yang digunakan masih sangat terbatas. Hal ini disebabkan teknologi perangkat keras maupun perangkat lunak masih sangat jauh jika dibandingkan dengan kondisi sekarang. Tujuan utama sistem informasi pada saat itu adalah untuk melakukan otomatisasi proses bisnis yang berjalan pada organisasi.

Pada periode sekitar tahun 1970an, sistem informasi sudah lebih berkembang. Perkembangan sistem informasi saat itu didominasi dari sudut pandang data. Teknologi basis data saat itu berkembang cukup pesat. Jadi, fokus utama sistem informasi saat itu adalah penyimpanan dan pengaksesan data. Pada saat itu sistem informasi biasanya masih digunakan pada suatu bagian organisasi, khususnya bagian keuangan. Oleh karena itu, kita sekarang sering kali melihat pada suatu organisasi, departemen/bagian sistem informasi (kadang juga disebut bagian teknologi informasi) berada di bawah departemen keuangan.

Pada periode tahun 1980an, sistem informasi berkembang lebih ke arah CSCW (*Computer Support Cooperative Work*). CSCW adalah aplikasi yang mendukung kerjasama dalam organisasi, misalnya pemanfaatan email, dokumen editor, dan lain-lain. Pada periode ini, sistem informasi mulai mengarah ke bentuk *client server*. Selain itu, pada periode ini pemanfaatan sistem informasi sudah mulai bertambah luas. Sistem informasi sudah dimanfaatkan pada bermacam-macam bagian organisasi, misalnya bagian keuangan, sumber daya manusia, pemasaran, dan lain-lain.

Pada tahun 1990an, internet berkembang sangat cepat. Perkembangan tersebut juga mendorong perkembangan sistem informasi. Sistem informasi mulai dimanfaatkan teknologi internet maupun teknologi web. Pada saat itu usaha untuk membuat suatu sistem informasi yang terintegrasi untuk seluruh organisasi sudah mulai dilakukan. Perusahaan-perusahaan perangkat lunak besar di dunia juga mulai mengembangkan sistem informasi yang disesuaikan dengan *best practice* yang ada, misalnya aplikasi ERP (*Enterprise Resource Planning*), CRM (*Customer Relationship Management*), SCM (*Supply Chain Management*), dan lain-lain.

Pada tahun 2000an, sistem informasi berkembang semakin pesat. Perkembangan ini didorong dengan semakin berkembangan teknologi internet, dengan kapasitas semakin besar dan harga yang semakin murah. Sudah banyak organisasi yang telah mengintegrasikan sistem informasi mereka dengan sistem informasi organisasi lain untuk mendukung kegiatan organisasi tersebut.

Pada masa mendatang, sistem informasi akan semakin berkembang lagi. Perkembangan teknologi dan perubahan dunia usaha yang sangat cepat, mendorong organisasi untuk mengembangkan suatu sistem informasi yang mampu beradaptasi dengan cepat menghadapi perubahan tersebut. Sistem informasi tersebut juga harus dapat diintegrasikan dengan bermacam-macam sistem yang lain agar kinerja organisasi menjadi lebih efisien.

1.2 *Definisi Pengembangan Sistem Informasi*

Pengembangan sistem informasi ialah satu set aktivitas, metode, praktik terbaik, siap dikirimkan, dan peralatan terotomasi yang digunakan oleh stakeholder untuk mengembangkan dan memelihara sistem informasi dan perangkat lunak.

Biasanya pengembangan sistem dilakukan apabila sistem yang lama sudah tidak bisa mengimbangi/memadai kebutuhan atau pun perkembangan perusahaan, sehingga terdapat beberapa pendapat tentang definisi pengembangan sistem, antara lain:

- Menyusun suatu sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang telah ada.
- Suatu proses pengaplikasian teknologi informasi untuk suatu tujuan tertentu atau menyelesaikan suatu masalah.
- Memilah suatu masalah yang besar dan kompleks menjadi beberapa bagian kecil yang dapat dikelola.

Pengembangan sistem dapat berarti menyusun suatu sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan/memperbaiki sistem yang telah ada.

Dengan telah dikembangkannya sistem yang baru, maka diharapkan akan terjadi peningkatan-peningkatan di sistem yang baru, yaitu meningkatkan:

- *Performance* (kinerja),` peningkatan terhadap kinerja sistem yang baru sehingga menjadi lebih efektif. Kinerja dapat diukur dari *throughput* (jumlah dari pekerjaan yang dapat dilakukan suatu saat tertentu dan *response time* (rata-rata waktu yang tertunda diantara dua transaksi/pekerjaan ditambah dengan waktu response untuk menanggapi pekerjaan tersebut).
- *Information* (informasi), peningkatan terhadap kualitas informasi yang disajikan.
- *Economy* (ekonomis), peningkatan terhadap manfaat-manfaat/keuntungan-keuntungan/penurunan-penurunan biaya yang terjadi.
- *Control* (pengendalian), peningkatan terhadap pengendalian untuk mendeteksi dan memperbaiki kesalahan-kesalahan serta kecurangan-kecurangan yang dan akan terjadi.
- *Efficiency* (efisiensi), peningkatan terhadap efisiensi operasi.
- *Services* (pelayanan), peningkatan terhadap pelayanan yang diberikan oleh sistem.

1.3 *Prinsip dan Perlunya Pengembangan Sistem Informasi*

1.3. 1 Prinsip Pengembangan Sistem Informasi

Beberapa prinsip yang harus digunakan pada saat pengembangan sistem adalah:

- Prinsip - 1 : Libatkan para pengguna sistem
  Guna menghindari konflik antara pengguna dan pengembang sistem, maka dalam menciptakan solusi dengan teknologi yang menarik harus melibatkan pengguna sistem yang mengetahui masalah-masalah organisasi yang sebenarnya. Hal ini dilakukan karena tujuan akhir dari pengembangan sistem ini adalah mendukung kebutuhan yang diperlukan oleh pihak manajemen.
- Prinsip – 2 : Gunakan pendekatan pemecahan masalah
  Pendekatan pemecahan masalah yang klasik adalah:
  - Mempelajari dan memahami masalah, konteks dan pengaruhnya.
  - Mendefinisikan persyaratan yang harus dipenuhi oleh semua solusi.
  - Mengidentifikasikan solusi-solusi calon yang memenuhi persyaratan dan memilih solusi terbaik.
  - Merancang dan atau mengimplementasikan solusi terpilih.

- Mengamati dan mengawasi pengaruh solusi dan memperbaiki solusi tersebut.

Analis sistem harus mendekati semua proyek dengan menggunakan beberapa variasi pendekatan pemecahan masalah tersebut.

Prinsip – 3 : Bentuklah fase dan aktivitas

Fase-fase yang dapat dibentuk dalam pengembangan sistem adalah definisikan lingkup, analisis masalah, analisis persyaratan, desain logis, analisis keputusan, desain fisik dan integrasi, konstruksi dan pengujian serta instalasi dan pengujian.

Prinsip – 4 : Dokumentasikan sepanjang pengembangan

Dokumentasi sangat berguna untuk pengembangan sistem berikutnya. Dokumentasi seharusnya dilakukan dari awal pengembangan sistem sampai proses tersebut selesai dilakukan.

Prinsip – 5 : Bentuklah Standar

Untuk mencapai atau memperbaiki integrasi sistem, organisasi beralih ke standar-standar yang berbentuk arsitektur teknologi informasi enterprise. Contoh standarnya adalah:

- Teknologi *database – engine*
- Teknologi perangkat lunak
- Teknologi antarmuka

Prinsip – 6 : Kelola proses dan proyek

- Manajemen proses adalah aktivitas terus-menerus yang mendokumentasikan, mengajarkan, mengawasi penggunaan, dan memperbaiki metodologi („proses") terpilih organisasi untuk pengembangan sistem. Manajemen proses peduli dengan fase, aktivitas, barang siap dikirim, dan standar kualitas yang seharusnya diterapkan secara konsisten ke semua proyek.
- Manajemen proyek adalah proses pelingkupan, perencanaan, penyediaan staf, pengorganisasian, pengarahan, dan pengontrolan sebuah proyek untuk mengembangkan sebuah sistem informasi dengan biaya minimal, dalam keragka waktu yang ditentukan dan dengan kualitas yang dapat diterima.

Prinsip – 7 : Membenarkan sistem informasi sebagai investasi modal

Pengembangan suatu sistem tentu memerlukan modal yang besar sehingga pengembangan sistem juga merupakan sebuah investasi untuk perusahaan itu sendiri. Beberapa hal yang harus diperhatikan terhadap investasi modal adalah semua alternatif yang ada harus diinvestigasi, dan investasi yang terbaik harus bernilai. Hasil yang diperoleh dengan menyeimbangkan biaya seumur hidup pengembangan, perawatan dan

pengoperasian sebuah sistem informasi dan keuntungan-keuntungan yang diperoleh dari sistem itu.

- Prinsip – 8 : Janganlah takut untuk membatalkan atau merevisi lingkup
  Pendekatan *creeping commitment* dapat dilakukan untuk merevisi lingkup, yaitu strategi tempat kepraktisan dan risiko dievaluasi ulang secara berkesinambungan melalui sebuah proyek. Anggaran dan tenggat waktu proyek disesuaikan. Mendefinisikan bagaimana tiap unit bisnis akan berkontribusi pada rencana enterprise.
- Prinsip – 9 : Bagilah dan taklukkan
  Dalam analisis sistem, prinsip ini sering disebut *factoring*, yaitu dengan berulang-ulang membagi masalah yang lebih besar (sistem) kedalam bagian-bagian (subsistem) yang lebih mudah dikelola, menyederhanakan proses pemecahan masalah.
- Prinsip – 10 : Desainlah sistem untuk pertumbuhan dan perubahan
  Bisnis-bisnis berubah setiap waktu, kebutuhan berubah, prioritas juga berubah. Untuk alasan ini maka metodologi yang baik harus mencakup kenyataan perubahan. Sistem harus didesain untuk mengakomodasi persyaratan-persyaratan pertumbuhan dan perubahan.

1.3.2 Perlunya Pengembangan Sistem Informasi

Dengan seiringnya perkembangan jaman maka sebuah sistem tentu tidak selamanya dapat digunakan dengan baik. Untuk itu perlu ada perubahan terhadap sistem tersebut baik dengan cara memperbaiki sistem yang lama atau pun jika perlu untuk mengganti sistem yang lama. Ada beberapa hal yang mendasari hal tersebut, antara lain:

- Ada permasalahan pada sistem yang lama.
  Permasalahan yang dimaksud disini seperti adanya ketidakberesan pada sistem yang lama sehingga hasilnya pun tidak sesuai dengan yang diharapkan. Contohnya: terdapat kesalahan-kesalahan baik yang disengaja atau pun tidak yang menyebabkan data pada suatu perusahaan tidak dapat terjamin kebenarannya, adanya kesempatan atau peluang anggota dari sistem tersebut untuk melakukan kecurangan. Permasalahan yang lain juga dapat disebabkan oleh pertumbuhan organisasi tersebut. Contohnya: pada sebuah perusahaan perdagangan yang berkembang yang sebelumnya hanya sebatas dalam kota, kini tumbuh hingga skala nasional bahkan internasional. Pertumbuhan organisasi (perusahaan) memaksa sistem yang dimiliki sebelumnya harus disesuaikan dengan kebutuhan kerja dari

perusahaan tersebut, misalnya transaksi yang sebelumnya bersifat konvensional kini lebih modern dengan memanfaatkan internet.

- Untuk meraih kesempatan (*opportunities*).
  Sebuah sistem harus diperbaiki atau dikembangkan juga disebabkan untuk meraih kesempatan dari suatu organisasi atau perusahaan. Misalnya pada tingkat manajer pada sebuah perusahaan dituntut untuk cepat menghasilkan suatu kebijakan agar perusahaan mendapatkan keuntungan yang lebih banyak, sehingga perusahaan tersebut memanfaatkan Sistem Pendukung Keputusan agar kebijakan yang didapat lebih cepat.
- Adanya instruksi-instruksi (*directives*).
  Sistem harus diperbaharui atau dikembangkan juga disebabkan oleh faktor eksternal seperti pemerintah. Adanya kebijakan-kebijakan pemerintah memaksa sebuah perusahaan menggunakan sistem yang tidak bertentangan dengan kebijakan tersebut.

Pengembangan atau pembuatan sebuah sistem tentu tidak memakan biaya yang sedikit, sehingga organisasi harus secara bijak menentukan apakah sistem yang digunakan masih layak untuk dipakai atau sudah harus dikembangkan atau diganti. Indikator-indikator yang menyebabkan sistem yang lama harus diperbaiki, ditingkatkan bahkan diganti keseluruhannya adalah adanya:

- keluhan dari pelanggan
- pengiriman barang yang sering tertunda
- pembayaran gaji yang terlambat
- laporan yang tidak tepat waktu
- isi laporan yang (sering) salah
- tanggung jawab yang tidak jelas
- waktu kerja yang berlebihan
- ketidakberesan kas
- produktivitas tenaga kerja yang rendah
- banyak pekerja yang menganggur
- kegiatan yang tumpang tindih
- tanggapan yang lambat terhadap konsumen
- kehilangan kesempatan kompetisi pasar
- kesalahan-kesalahan manual yang tinggi
- persediaan barang yang terlalu tinggi
- pemesanan kembali barang yang tidak efisien
- biaya operasi yang tinggi
- file-file yang kurang teratur

- keluhan dari supplier karena tertundanya pembayaran
- tumpukan *back-order* (tertundanya pengiriman karena kurangnya persediaan barang)
- investasi yang tidak efisien
- peramalan penjualan dan produksi tidak tepat
- kapasitas produksi yang menganggur (*idle capacities*)
- pekerjaan manajer yang terlalu praktis.

Pengembangan sistem informasi dilakukan untuk mendukung kegiatan bisnis dalam organisasi, tahapannya terdiri dari inisialisasi, analisis, desain, dan implementasi. Pengembangan sistem informasi dapat berupa pembuatan suatu sistem baru maupun penambahan atau perubahan modul pada sistem yang sudah ada. Secara umum, alur pengembangan suatu sistem informasi mempunyai beberapa tahapan. Tahapan pengembangan sistem informasi sering kali disebut juga sebagai *System Development Life Cycle* (SDLC).

Dalam pengembangan sistem informasi, terdapat 2 (dua) hal utama yang harus diperhatikan.

- Produk. Produk adalah produk yang harus dihasilkan pada setiap tahap pengembangan sistem informasi. Kesalahan dalam pembuatan produk dalam setiap tahap akan menyebabkan kesalahan yang semakin besar pada produk akhir.
- Proses. Proses adalah proses pengembangan sistem informasi. Proses ini meliputi tahapan pengembangan mulai dari tahap *feasibility* sampai *implementation*. Jika proses tersebut tidak dilaksanakan sesuai dengan jadwal maka kemungkinan kegagalan proyek menjadi semakin besar.

1.4 *Tim Pengembang Sistem Informasi*

Suatu proyek pengembangan sistem informasi biasanya dikembangkan oleh sebuah tim. Tim tersebut biasanya terdiri dari beberapa posisi sebagai berikut:

- *Project Leader* yaitu penanggung jawab utama proyek pengembangan sistem informasi. Seorang *project leader* harus mampu mengatur waktu dan sumber daya agar sistem informasi dapat diselesaikan sesuai dengan target yang telah ditetapkan. Dalam sebuah proyek pengembangan sistem informasi, seorang *project leader* sebaiknya tidak merangkap jabatan lain untuk menghindari adanya konflik kepentingan.

- *System Analyst* yaitu orang yang bertugas untuk melakukan analisis terhadap kebutuhan *user* dan kemudian mendokumentasikan kebutuhan *user* tersebut dalam suatu dokumen teknis yang mudah dipahami oleh anggota tim pengembangan sistem informasi. Seorang *system analyst* yang baik sebaiknya mempunyai pengetahuan dibidang sistem informasi dan pengembangan perangkat lunak sehingga dia mampu merepresentasikan kebutuhan *user* dengan baik dalam suatu dokumen. Selain itu, *system analyst* juga dituntut untuk mempunyai pengetahuan umum yang luas agar mempermudah dalam memahami kebutuhan *user*.
- *System Designer* yaitu orang yang bertugas untuk mendesain sistem berdasarkan dokumen kebutuhan *user*.
- *Programmer* yaitu orang yang bertugas untuk mengimplementasikan desain tersebut menjadi kode program.
- *Software Quality Assurance* (SQA) yaitu orang yang bertugas untuk memastikan semua proses pengembangan sistem informasi berjalan dengan baik dan memastikan produk yang dihasilkan sesuai yang diharapkan.

1.5 *Pendekatan dan Metodologi Pengembangan Sistem*

1.5.1 Pendekatan Pengembangan Sistem

Terdapat beberapa pendekatan yang digunakan untuk pengembangan sistem dan dapat dilihat dari beberapa sudut pandang, antara lain:

- Metodologi yang digunakan:
  - Pendekatan klasik: pendekatan di dalam pengembangan sistem mengikuti tahapan daur/siklus hidup sistem tanpa dibekali alat-alat dan teknik-teknik yang memadai. Permasalahan yang akan timbul antara lain pengembangan *software* akan sulit, biaya perawatan dan pemeliharaan mahal, kemungkinan kesalahan sistem besar dan keberhasilan sistem kurang terjamin.
  - Pendekatan terstruktur: pendekatan di dalam pengembangan sistem mengikuti tahapan daur/siklus hidup sistem dan dibekali alat-alat dan teknik-teknik yang memadai.

- Sasaran yang ingin dicapai:
  - Pendekatan sepotong: pendekatan di dalam pengembangan sistem yang menekankan pada suatu kegiatan atau aplikasi tertentu saja. Dilihat hanya pada sasaran aplikasi saja.
  - Pendekatan sistem: pendekatan ini memperhatikan sistem informasi sebagai satu kesatuan yang terintegrasi untuk masing-masing kegiatan atau aplikasinya.
- Cara menentukan kebutuhan dari sistem:
  - Pendekatan bawah-naik (*bottom – up*), dalam pendekatan ini dilakukan perumusan untuk menangani transaksi dan naik ke level atas dengan merumuskan kebutuhan informasi berdasarkan pada transaksinya.
  - Pendekatan atas-turun(*top – down*), pendekatan ini mulai mendefinisikan sasaran dan kebijaksanaan organisasi.
- Cara mengembangkannya:
  - Pendekatan sistem-menyeluruh, pendekatan yang mengembangkan sistem secara serentak dan menyeluruh.
  - Pendekatan moduler, pendekatan yang memecah sistem yang rumit menjadi beberapa bagian atau modul yang lebih sederhana.
- Teknologi yang digunakan:
  - Pendekatan lompatan jauh (*great loop approach*), menerapkan perubahan secara menyeluruh dengan serentak menggunakan teknologi canggih.
  - Pendekatan berkembang (*evolutionary approach*), pendekatan yang menggunakan teknologi canggih hanya untuk aplikasi-aplikasi yang memerlukan saja pada saat itu dan akan terus berkembang dengan mengikuti kebutuhan.

1.5.2 Metodologi Pengembangan Sistem

Metodologi adalah kesatuan metode-metode, prosedur-prosedur, konsep pekerjaan, aturan yang digunakan oleh suatu ilmu pengetahuan, seni dan disiplin ilmu lainnya.

Metode adalah aturan, cara, teknik yang sistematik untuk mengerjakan sesuatu.

Metodologi pengembangan sistem adalah metode-metode, prosedur-prosedur yang digunakan untuk melakukan pengembangan sistem informasi.

Terdapat macam-macam representasi metodologi pengembangan sistem, yaitu:

- *Architected Rapid*
- *Application Development (Architected RAD)*
- *Dynamic Systems Development Methodology (DSDM)*
- *Joint Application Development (JAD)*
- *Information Engineering (IE)*
- *Rapid Application Development (RAD)*
- *Rational Unified Process (RUP)*
- *Structured Analysis and Design (SAD)*
- *eXtreme Programming (XP)*

Dalam pengembangan sistem informasi, penjadwalan proses merupakan hal yang harus diperhatikan dengan serius. Jika sejak awal kita telah gagal dalam membuat jadwal yang baik, maka dapat dipastikan proyek tersebut akan kacau sehingga mengakibatkan molornya waktu proyek dan membengkaknya biaya.

Ketika dilihat sekilas, penjadwalan seperti Gambar 2 di bawah terlihat cukup baik. Tetapi jika diperhatikan lebih lanjut, Gambar 2 menunjukkan sebuah penjadwalan yang tidak realistis. Gambar 3 menunjukkan penjadwalan yang lebih realistis. Berikut ini adalah beberapa penyebab Gambar 2 tidak realistis.

- Kegiatan *feasibility* hanya dilakukan pada saat awal proyek dan kegiatan *analysis* belum dikerjakan sama sekali. *Feasibility* sebaiknya dilakukan kembali setelah melakukan *analysis* agar analisis resiko menjadi lebih akurat.
- Hasil *design* khususnya desain antarmuka dan desain interaksi sebaiknya diajukan ke *user* karena tidak mungkin hanya dengan satu kali proses analisis, tanpa memberikan *prototype*, akan menghasilkan sesuai dengan keinginan *user*.
- Setelah proses *quality assurance* tidak ada lagi proses *development*. Hal ini tidak mungkin karena setiap aplikasi pasti terdapat kesalahan-kesalahan yang harus diperbaiki. Dan tujuan kegiatan *quality assurance* adalah menemukan kesalahan-kesalahan tersebut untuk kemudian diperbaiki sebelum masuk ke tahap *implementation.*

Aktivitas	Maret	April	Mei		Juni	Juli	Agustus
Feasibility							
Analysis							
Design							
Development							
Quality Assurance							
Implementation							

Gambar 2 Penjadwalan Tidak Realistis

Aktivitas	Maret	April	Mei		Juni	Juli	Agustus
Feasibility							
Analysis							
Design							
Development							
Quality Assurance							
Implementation							

Gambar 3 Penjadwalan Realistis

1.6 *Pengertian* System Development Life Cycle *(SDLC)*

System Development Life Cycle disingkat dengan SDLC. SDLC merupakan siklus pengembangan sistem. Pengembangan sistem teknik (*engineering system development*). SDLC berfungsi untuk menggambarkan tahapan-tahapan utama dan langkah-langkah dari setiap tahapan yang secara garis besar terbagi dalam empat kegiatan utama, yaitu *initiation*, *analysis, design* dan *implementation.* Setiap kegiatan dalam SDLC dapat dijelaskan melalui tujuan (*purpose*) dan hasil kegiatannya (*deliverable*). SDLC didefinisikan oleh Departemen Kehakiman AS sebagai sebuah proses pengembangan *software* yang digunakan oleh *analyst system*, untuk mengembangkan sebuah sistem informasi. SDLC mencakup kebutuhan (*requirement*), validasi, pelatihan, kepemilikan (*user ownership*) sebuah sistem informasi yang diperoleh melalui investigasi, analisis, desain, implementasi, dan perawatan *software. Software* yang dikembangkan berdasarkan SDLC akan menghasilkan sistem dengan kualitas yang tinggi, memenuhi harapan penggunanya, tepat dalam waktu dan biaya, bekerja dengan efektif dan efsien dalam infrastruktur teknologi informasi yang ada atau yang direncanakan, serta murah dalam perawatan dan pengembangan lebih lanjut.

1.7 *Sejarah Perkembangan SDLC*

Sejarah perkembangan *System Development Life Cycle* (SDLC) diawali pada pertengahan tahun 60-an dimana terjadi kegagalan yang sangat besar dalam penerapan aplikasi EDP (*Electronic Data Processing*) untuk sistem-sistem besar, sebagian besar disebabkan tidak adanya pengembangan sistem.

Sesudah terjadinya kegagalan tersebut pada akhir tahun 60-an dan awal 70-an, kesadaran akan pentingnya metodologi pengembangan sistem mulai tumbuh. Sejak itulah berbagai proposal metodologi mulai dibuat dan penerapan mulai terlihat. Para desainer dari hampir semua bidang metodologi pengembangan sistem informasi mempunyai pandangan yang sama, yaitu: mereka telah mengetahui bahwa proses pengembangan sistem informasi, baik yang berdasarkan komputer atau tidak, menyerupai dengan proses pengembangan *sistem engineering.*

Hubungan dengan konstruksi dan operasi berbagai jenis gedung, mesin, peralatan kimia yang merupakan contoh perkembangan sistem informasi *engineering*, kita dapat meringkas tahap-tahap proses secara umum perkembangan tersebut adalah perencanaan (*planning*), analisis (*analysis*), desain (*design*), pelaksanaan (*implementation*) dan perawatan (*maintenance*).

Dalam tahap perencanaan, kita mengumpulkan informasi tentang permasalahan serta persyaratannya. Kemudian kita menentukan kriteria dan pembatasan pemecahan, serta memberikan alternatif jalan keluarnya. Dalam tahap analisis, kita menguji alternatif pemecahan berdasarkan kriteria dan batasan-batasan. Analisis merupakan pusat dari semua proses perkembangan. Tahap berikutnya yaitu desain, dapat dikatakan sebagai hasil dari sistem baru. Tahap desain juga dapat dikatakan sebagai pemecahan yang optimum atas sejumlah kebutuhan penting dari suatu set pada keadaan khusus atau sebagai kegiatan kreativitas yang meliputi pembuatan barang baru dan berguna yang belum pernah ada sebelumnya. Sistem yang tersusun dibentuk dan dioperasikan. Perawatan dilakukan pada tiap sistem operasional.

Istilah daur/siklus hidup (*life cycle*) pada suatu sistem digunakan untuk menjelaskan tahap-tahap perkembangan sistem, serta langkah-langkah dalam proses perkembangannya. Untuk mengetahui proses sistem informasi dan proses sistem *engineering*, kita harus membandingkan daur/siklus hidup kedua sistem tersebut. Dengan mengetahui daur/siklus hidup sistem informasi tahun

1960 sampai dengan tahun 1983, kita akan mengetahui perbedaannya. Daur hidup sistem informasi sangat dekat dengan daur hidup yang terjadi dalam sistem *engineering*; perencanaan, analisis, desain, pelaksanaan, dan perawatan. Proses perkembangan sistem informasi merupakan proses *engineering*.

Meskipun selama hampir dua puluh tahun putaran sistem informasi, yang kurang lebih berisi langkah-langkah yang sama, namun pemberian nama dan dukungan pada langkah-langkah tersebut belum cukup untuk mengembangkan sistem informasi yang baik. Kekurangan tersebut adalah bahwa pada tiap perkembangan sistem *engineering* terdapat beberapa peralatan dan metodologi yang digunakan secara paralel dengan daur/siklus hidup sistem tersebut. Kegagalan dalam menentukan tuntutan dan peran serta pemakai dalam perkembangan sistem juga penyebab lain dari kegagalan sistem informasi, demikian juga masalah sulitnya memperoleh komputer dari produsen, staf yang tidak memenuhi syarat, batas waktu yang tidak realistis dan manajemen yang tidak memadai.

Kesalahan interpretasi mengenai tahap-tahap perkembangan sistem di atas adalah linier. Seolah olah semua fase dan tahap terlihat berderet secara berurutan. Tetapi sebenarnya tidak demikian. Semua tahap pada proses perkembangan sistem tersebut mempunyai sifat dasar yang iteratif yaitu pekerjaan pada suatu tahap sering harus diulang-ulang, dan apa pun yang dikerjakan pada suatu tahap mungkin perlu dikoreksi secara keseluruhan.

Meskipun terdapat beberapa variasi diantara masing-masing tahap, metode sistem klasik ternyata tidak cukup untuk menghasilkan sistem informasi yang baik, kemudian sebagai tambahan pada penamaan tahap-tahap dari suatu daur/siklus hidup sistem, kita harus mempunyai beberapa peralatan dan teknik baku untuk mengembangkan sistem tersebut.

1.8 *Tahapan* System Development Life Cycle *(SDLC)*

SDLC meliputi tahapan berikut:

- *System initiation* ialah perencanaan awal untuk sebuah proyek guna mendefinisikan lingkup, tujuan, jadwal dan anggaran bisnis awal yang diperlukan untuk memecahkan masalah atau kesempatan yang direpresentasikan oleh proyek. Lingkup proyek mendefinisikan area bisnis yang akan ditangani oleh proyek dan tujuan-tujuan yang akan dicapai. Lingkup dan tujuan pada akhirnya berpengaruh pada komitmen

sumber yaitu jadwal dan anggaran yang harus dibuat supaya berhasil menyelesaikan proyek.

- *System analysis* ialah studi domain masalah bisnis untuk merekomendasikan perbaikan dan menspesifikasikan persyaratan dan prioritas bisnis untuk solusi. Analisis system ditujukan untuk menyediakan tim proyek dengan pemahaman yang lebih menyeluruh terhadap masalah-masalah dan kebutuhan-kebutuhan yang memicu proyek. Area bisnis dipelajari dan dianalisis untuk memperoleh pemahaman yang lebih rinci mengenai apa yang bekerja, apa yang tidak bekerja dan apa yang dibutuhkan.
- *System design* ialah spesifikasi atau konstruksi solusi yang teknis dan berbasis komputer untuk persyaratan bisnis yang diidentifikasikan dalam analisis sistem. Selama desain sistem, pada awalnya akan mengekspolarasi solusi teknis alternatif. Setelah alternatif solusi disetujui, fase desain sistem mengembangkan cetak biru (*blueprint*) dan spesifikasi teknis yang dibutuhkan untuk mengimplementasikan database, program, antarmuka pengguna dan jaringan yang dibutuhkan untuk sistem informasi,
- *System implementation* ialah konstruksi, instalasi, pengujian dan pengiriman sistem ke dalam produksi (artinya operasi sehari-hari). Implementasi sistem mengontruksi sistem informasi baru dan menempatkannya ke dalam operasi, selanjutnya dilaksanakan pengujian.

Kuis Benar Salah

1. Sistem informasi dalam organisasi meng-*capture* dan mengelola data untuk menghasilkan informasi berguna yang mendukung organisasi dan karyawan, pelanggan, pemasok dan rekan kerjanya.
2. Salah satu tim pengembang sistem informasi adalah *system designer.*
3. Terdapat 9 (Sembilan) prinsip mendasar pengembangan sistem.
4. Salah satu prinsip mendasar pengembangan sistem adalah dokumentasi.
5. Dengan menentukan jadwal dan anggaran proyek pada lingkup dan tujuan awal artinya juga menentukan *baseline* (titik awal) dimana setiap stakeholderdapat menerima kenyataan bahwa semua perubahan yang terjadi tidak akan berpengaruh pada jadwal dan anggaran.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Seorang analis sistem harus mengembangkan dan memiliki keterampilan, pengetahuan dan sifat berikut, KECUALI:
   a. Pengetahuan kerja sistem informasi
   b. Karakter dan etika
   c. Pengetahuan tentang ilmu ekonomi
   d. Keterampilan pemecahan masalah umum
   e. Pengetahuan umum proses dan terminologi bisnis

2. Pekerja informasi adalah stakeholder dalam sistem informasi. Pekerja informasi termasuk orang-orang yang pekerjaannya melibatkan pembuatan, pengumpulan, pemrosesan, distribusi, dan penggunaan informasi. Mereka adalah:
   a. Pemilik sistem
   b. Pengguna sistem
   c. Desainer sistem
   d. Analis sistem
   e. Jawaban a, b, c dan d benar

3. Alasan pengembangan sistem informasi yang paling tepat adalah karena:
   a. adanya teknologi baru
   b. organisasi mendapatkan kerugian yang terus-menerus
   c. adanya kesalahan dalam pengelolaan manajemen pada organisasi
   d. mengganti sistem lama dan untuk meraih kesempatan
   e. mengharapkan keuntungan yang sebesar-besarnya

4. Di bawah ini adalah representasi metodologi pengembangan sistem, KECUALI:
   a. Application Development (Architected RAD)
   b. Framework for the Application of System Thinking (FAST)
   c. Dynamic Systems Development Methodology (DSDM)

d. eXtreme Programming (XP)

e. Information Engineering (IE)

5. Pengujian terhadap sistem dilaksanakan pada tahap mana dalam SDLC:
   a. System initiation
   b. System analysis
   c. System design
   d. System implementation
   e. Jawaban a, b, c dan d salah

Latihan

1. *Basic* (Pertanyaan yang jawabannya ada pada isi buku ini)
   a. Apakah definisi sistem, sistem informasi, dan teknologi informasi?
   b. Apa perbedaan sistem informasi dan teknologi informasi? Gambarkan keterhubungan antara sistem informasi dan teknologi informasi!
   c. Jelaskan mengenai sejarah perkembangan sistem informasi!
   d. Sebutkan faktor-faktor yang mempengaruhi perkembangan sistem informasi!
   e. Sebutkan tahap-tahap pengembangan sistem informasi!

2. *Advanced* (Pertanyaan terkait bab ini yang jawabannya harus dicari di luar buku ini)
   a. Apakah yang dimaksud dengan ERP, CRM, dan SCM? Jelaskan dan berikan contohnya!
   b. Faktor apa saja yang mempengaruhi penjadwalan proyek pengembangan sistem informasi?

2 Perencanaan Sistem

Overview

Perencanaan sistem merupakan tahap paling awal sebelum melakukan pengembangan sistem informasi. Tahap ini digunakan untuk menentukan apakah pengembangan sistem informasi akan dilakukan atau tidak. Pada bab ini dijabarkan mengenai urutan kegiatan yang dilakukan pada saat perencanaan sistem.

Tujuan

1. Mahasiswa mempunyai gambaran mengenai hal-hal yang biasanya dilakukan sebelum melakukan pengembangan sistem informasi.
2. Mahasiswa mampu melakukan perencanaan sistem khususnya sistem-sistem yang sederhana.
3. Mahasiswa mampu membuat dokumentasi perencanaan sistem.

2.1 *Definisi Perencanaan Sistem*

Perencanaan sistem atau *feasibility* adalah tahap pertama yang harus dilakukan sebelum mulai melakukan pengembangan sistem informasi. Terdapat beberapa hal yang sebaiknya dilakukan pada tahap ini, antara lain adalah mendefinisikan proyek, memodelkan proyek, membuat perkiraan anggaran dan penjadwalan proyek, menyeimbangkan rencana proyek dan menyetujui rencana proyek.

2.2 *Perlunya Perencanaan Sistem*

Perencanaan sistem merupakan suatu aktivitas yang harus dilaksanakan sebelum dikembangkannnya sebuah sistem. Perencanaan sistem perlu dilakukan agar pembangunan/pengembangan sistem sesuai *blueprint* yang ada, yang sesuai dengan visi, misi, tujuan dan sasaran organisasi. Biasanya pengembangan sistem dilaksanakan dalam lingkup proyek. Sebelum pelaksanaan proyek pengembangan sistem informasi dimulai, maka proyek tersebut harus mendapatkan persetujuan dari pengambil keputusan. Pengambil keputusan pada suatu organisasi yaitu manajemen tingkat atas (*executive*). Namun, kadang-kadang manajemen akan meminta pendapat bawahannya, manajer level menengah (*middle manager*) maupun calon pengguna aplikasi (*functional user*), dalam melakukan pengambilan keputusan pelaksanaan proyek.

Oleh karena itu, dalam melakukan pendefinisian proyek, anda harus memahami karakteristik kebutuhan para pengambil keputusan. Berikut ini adalah karakteristik umum mengenai orang-orang yang terlibat pengambilan keputusan tersebut.

- *Executive* (manajemen tingkat atas)
  Prioritas utama *executive* adalah ROI (*Return On Invesment*). Jadi agar proyek dapat disetujui, maka anda harus mampu meyakinkan mereka bahwa proyek tersebut dapat meningkatkan ROI.
- *Middle manager* (manajer level menengah)
  Prioritas utama *middle manager* biasanya adalah bagaimana meningkatkan produktivitas kerja. Jadi sistem informasi yang akan dikembangkan tersebut harus mampu menunjukkan seberapa besar produktivitas kerja akan meningkat dengan adanya sistem baru tersebut.

- *Functional user* (pengguna aplikasi langsung)
  Kebutuhan utama *functional user* adalah suatu aplikasi yang akan mempermudah pekerjaan mereka. Jadi jika *functional user* dilibatkan dalam pengambilan keputusan, maka anda harus mampu menunjukkan kemudahan-kemudahan apa yang akan diperoleh *functional user* dengan adanya sistem informasi yang akan dikembangkan tersebut.

Pada tahap ini, dokumen yang dihasilkan adalah dokumen proposal proyek. Sebuah dokumen proposal proyek tersebut minimal terdiri dari hal-hal sebagai berikut:

- Keuntungan yang akan diperoleh calon pengguna dengan adanya sistem informasi yang akan dikembangkan tersebut. Anda sebaiknya mengetahui siapa yang mengambil keputusan pengadaan sistem baru dan tunjukkan kelebihan sistem baru tersebut sesuai dengan karakteristik kebutuhannya.
- Rencana biaya yang dibutuhkan untuk pengembangan, jika anda menjual sistem informasi tersebut ke pihak lain, berarti rencana biaya pengembangan di sini diganti dengan harga sistem informasi yang anda jual.
- Waktu yang dibutuhkan untuk pengembangan sistem.

2.3 *Proses Perencanaan Sistem*

Proses perencanaan sistem dilakukan dengan:

- menetapkan suatu kerangka kerja strategi menyeluruh untuk memenuhi kebutuhan informasi pemakai.
- melibatkan manajer senior, pemakai senior dan profesional sistem.
  memastikan bahwa proyek yang diusulkan dievaluasi dan diprioritaskan.
  memenuhi alasan untuk melakukan perencanaan sistem:
  - dihubungkan dengan rencana bisnis
  - menghindari sejumlah kerugian
- membagi tugas dan tanggung jawab pada orang yang merencanakan sistem:
  - *Steering Committee* (SC), *Chief Information Officer* (CIO), *Chief Executive Officer* (CEO), *Chief Financial Officer* (CFO) dan Eksekutif Senior.
  - Tugas SC : merupakan penghubung antara tujuan bisnis dan sistem informasi yang membantu untuk mencapai tujuan tersebut.

- membuat komponen laporan:
  - komponen keseluruhan berhubungan dengan sumber daya yg akan diperoleh (3-5 tahun), meliputi : personil baru, *hardware, software*, peralatan telekomunikasi, lokasi computer dan keamanan.
  - komponen aplikasi: suatu portfolio yang disetujui dari proposal proyek sistem, secara luas menyatakan apa saja yang termasuk dalam komponen keseluruhan.
- melakukan komunikasi dengan analis sistem
  - keduanya berhubungan dengan proses mendefinisikan kebutuhan pemakai
  - perbedaannya pada cakupan dan tahap rinci
- memastikan bahwa pada perencanaan sistem, suatu sistem yang diusulkan harus layak dan mendukung faktor strategik. Untuk menilai kedua kemungkinan tersebut maka harus diadakan evaluasi terhadap faktor kelayakan dan faktor strategi.

2.4 Pemodelan Proyek

Pemodelan proyek mempunyai fokus pada pembuatan simulasi mengenai usaha yang dibutuhkan untuk mencapai tujuan proyek. Pemodelan ini menghasilkan sebuah WBS (*Work Breakdown Structure*) yang digunakan untuk menentukan semua usaha yang dibutuhkan untuk menyelesaikan proyek dengan sukses. WBS adalah daftar semua pekerjaan yang harus dilakukan untuk menghasilkan produk yang diinginkan.

Dalam sebuah proyek, ada banyak pekerjaan yang harus dilakukan. Sebuah pekerjaan yang kompleks, sebaiknya dipecah lagi menjadi beberapa sub-pekerjaan. Dan beberapa pekerjaan yang terlalu kecil dan detail sebaiknya digabungkan menjadi sebuah pekerjaan. Dalam menentukan pekerjaan apa saja yang harus dilakukan dalam sebuah proyek, agar tidak ada yang terlalu kompleks maupun terlalu detail, sangat tergantung dari pengalaman seseorang dan besar atau kecilnya proyek.

Pembuatan WBS membutuhkan kontribusi dari anggota tim yang akan terlibat dalam proyek tersebut. Sebuah metode yang efektif dalam pembuatan WBS adalah membuat sebuah sesi diskusi yang melibatkan semua anggota tim dan memberikan kesempatan bagi mereka untuk memberikan ide-ide yang mereka miliki. Setelah WBS selesai dibuat, tim tersebut kemudian harus menggambarkan keterhubungan antara setiap tugas pekerjaan, menentukan

tugas apa yang harus sudah selesai sebelum tugas lain dilakukan. Keterhubungan antar pekerjaan ini nanti dibutuhkan dalam melakukan proses penjadwalan.

Perkiraan dan penjadwalan proyek ini fokus kepada penentuan waktu, biaya, dan sumber daya yang dibutuhkan dalam pelaksanaan proyek. Kebanyakan orang yang melakukan estimasi, biasanya mulai melakukan estimasi dengan cara menentukan seberapa besar *man-hours* atau *man-days* yang dibutuhkan untuk menyelesaikan pekerjaan. Angka ini nanti juga dibutuhkan dalam menentukan waktu dan biaya yang dibutuhkan.

Berikut ini adalah tujuh tahapan proses estimasi.

- Langkah 1: Membuat estimasi pekerjaan
  Estimasi pekerjaan seharusnya melibatkan anggota tim yang menjalankan pekerjaan tersebut. Sehingga estimasi tersebut akan realistis dan anggota tim akan punya komitmen dan termotivasi untuk mencapai estimasi tersebut. Estimasi ini kemudian dapat dimodifikasi untuk menyesuaikan dengan jadwal dan sumber daya yang ada.
- Langkah 2: Membuat perencanaan awal
  Perencanaan awal proyek berisi sebuah jadwal yang dibuat berdasarkan ketergantungan antar pekerjaan (*task*) dan estimasi pekerjaan tersebut. Jadwal tersebut berisi kapan pekerjaan dimulai, berapa lama, dan kapan pekerjaan tersebut harus sudah selesai. Biaya dapat dihitung dari pekerjaan apa saja yang harus dilakukan dan biaya untuk pembelian barang.
- Langkah 3: Membandingkan perencanaan awal dengan tujuan
  Tahap selanjutnya adalah pembandingkan antara tujuan awal proyek dengan estimasi rencana jadwal dan biaya yang sudah dilakukan. Tujuan awal proyek biasanya merupakan hal yang konstan dan telah disetujui oleh *executive*. Negosiasi ini tidak diperlukan jika tujuan awal telah sesuai dengan rencana jadwal dan biaya yang dilakukan. Tetapi jika tidak sesuai, maka langkah 4, 5, 6 harus dilakukan.
- Langkah 4 : Negosiasi perubahan untuk estimasi
  Anda melakukan perubahan estimasi mengenai rencana waktu dan anggaran agar sesuai dengan tujuan awal. Langkah ini mengandung risiko sangat besar apabila anda melakukannya tanpa persetujuan anggota tim yang lain, maka anda akan kehilangan komitmen dan motivasi anggota tim. Anggota tim akan beranggapan jadwal dan anggarannya tidak realistis, sehingga kemungkinan proyek gagal menjadi sangat besar.

- Langkah 5 : Negosiasi perubahan untuk tujuan proyek
  Langkah ini adalah melakukan negosiasi dengan *executive* karena dengan perubahan estimasi yang telah anda lakukan, rencana awal tersebut tidak realistis. Perubahan rencana tersebut dapat berupa penambahan waktu dan anggaran maupun pengurangan kompleksitas sistem. Usahakan agar sebisa mungkin rencana yang anda lakukan telah disetujui oleh anda, anggota tim anda, dan *executive*.
- Langkah 6 : Membuat keputusan terus/berhenti
  Setelah melakukan langkah 4 dan 5, anda harus mengambil keputusan apakah akan meneruskan proyek tersebut maupun tidak.
- Langkah 7 : Mempersiapkan jadwal dan anggaran
  Rencana awal pengembangan sistem informasi telah siap. Rencana ini terdiri dari tiga hal, yaitu jadwal kegiatan (waktu mulai, durasi, dan waktu selesai), alokasi sumber daya manusia terhadap kegiatan, dan rencana anggaran.

Setelah rencana anggaran dan jadwal selesai dibuat, hal yang harus dilakukan selanjutnya adalah menyeimbangkan rencana proyek tersebut dengan kondisi organisasi. Biasanya sebuah organisasi akan menjalankan beberapa proyek. Dan dalam organisasi tersebut uang dan sumber daya manusia merupakan hal yang terbatas. Jadi hal yang harus dilakukan adalah mengatur jadwal dan anggaran agar semua proyek yang sedang dikerjakan dapat berjalan dengan baik. Terdapat bermacam-macam perangkat lunak yang dapat digunakan untuk membantu mengelola sumber daya tersebut. Dengan pengelolaan yang baik, maka sumber daya uang dan manusia dapat tersedia pada saat dibutuhkan.

Setelah rencana anggaran dan jadwal selesai dibuat, hal yang harus dilakukan selanjutnya adalah menyeimbangkan rencana proyek tersebut dengan kondisi organisasi. Biasanya sebuah organisasi akan menjalankan beberapa proyek. Dan dalam organisasi tersebut uang dan sumber daya manusia merupakan hal yang terbatas. Jadi hal yang harus dilakukan adalah mengatur jadwal dan anggaran agar semua proyek yang sedang dikerjakan dapat berjalan dengan baik.

Terdapat bermacam-macam perangkat lunak yang dapat digunakan untuk membantu mengelola sumber daya tersebut. Dengan pengelolaan yang baik, maka sumber daya uang dan manusia dapat tersedia pada saat dibutuhkan.

Tahap terakhir adalah persetujuan rencana, dokumen terkait rencana target (target tanggal selesai, target biaya, target rencana penggunaan sumber daya). Dokumen ini merupakan dokumen persetujuan antara *project leader*, *executive*, dan *client* yang akan digunakan sebagai acuan jika selama proyek berlangsung terjadi perubahan ruang lingkup proyek dan juga digunakan sebagai acuan mengukur performa tim.

Kuis Benar Salah

1. Perencanaan sistem merupakan tahap awal pengembangan sistem informasi.
2. Perencanaan sistem dilakukan dengan menetapkan suatu kerangka kerja strategi menyeluruh untuk memenuhi kebutuhan informasi pemakai.
3. Ada enam langkah untuk tahapan proses estimasi.
4. Salah satu tahapan dalam proses estimasi adalah memutuskan untuk terus melanjutkan proyek atau proyek dihentikan.
5. Sebuah WBS (*Work Breakdown Strucure*) adalah penguraian hierarchies proyek menjadi tugas-tugas dan sub-sub tugas. Beberapa tugas mewakili penyelesaian *milestone* atau penyelesaian produk-produk jadi selama proyek.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Proses perencanaan sistem dilakukan dengan:
   a. menetapkan suatu kerangka kerja strategi menyeluruh untuk memenuhi kebutuhan informasi pemakai.
   b. melibatkan manajer senior, pemakai senior dan profesional sistem.
   c. memastikan bahwa proyek yang diusulkan.
   d. memastikan bahwa proyek dievaluasi dan diprioritaskan.
   e. Semua jawaban benar.

2. Yang bertanggung jawab dalam perencanaan sistem adalah:
   a. *Steering Committee* (SC)
   b. *Chief Information Officer* (CIO)
   c. *Chief Executive Officer* (CEO)
   d. *Chief Financial Officer* (CFO)
   e. Semua jawaban benar

3. Setiap proyek pengembangan sistem sifatnya unik, maksudnya adalah:
   a. mensyaratkan keterampilan dan persyaratan tertentu yang kompeten.
   b. mendefinisikan aktivitas-aktivitas mana yang bersifat wajib dan opsional.
   c. menyatakan bahwa ia berbeda dari tiap proyek pengembangan sistem lain yang mendahuluinya.
   d. sesuai dengan spesifikasi yang telah ditentukan pada tahap perencanaan.
   e. mengembangkan sistem secara berurutan.

4. Sebuah proyek pengembangan sistem informasi dikatakan sukses, KECUALI:
   a. Sistem informasi yang dihasilkan diterima oleh pelanggan
   b. Sistem dikirimkan tepat waktu

c. Sistem dikirimkan sesuai dengan anggaran
d. Proses pengembangan sistem mempunyai pengaruh minimal pada operasi bisnis organisasi secara berkesinambungan
e. Manajer merasa puas karena biaya dapat diminimalisir untuk pengembangan sistem dengan membuang/melewati beberapa tahapan proses perencanaan sistem.

5. Di bawah ini adalah perangkat lunak manajemen proyek:
   a. *Microsoft Project*
   b. *Artemis Management System* 7000 dan 9000
   c. *Project Management (Result Management Suite)*
   d. *Primavera Project Planner dan Monte Carlo*
   e. Jawaban di atas benar semua

Latihan

1. *Basic*
   a. Apa akibatnya jika pengembangan sistem informasi tidak melalui tahap perencanaan sistem?
   b. Sebutkan langkah-langkah perencanaan sistem!
   c. Apakah yang dimaksud dengan WBS?

2. *Advanced*
   a. Apakah yang dimaksud dengan ROI (*Return On Investment*) dan mengapa banyak *executive* sangat memperhatikan ROI?
   b. Buatlah sebuah proposal proyek pengembangan sistem informasi sebuah apotek!
   c. Buatlah WBS untuk proyek pengembangan sistem informasi sebuah apotek!

3 Analisis Sistem

Overview

Bab ini menjelaskan mengenai kegiatan analisis sistem. Batasan mengenai hal-hal apa saja yang dilakukan pada tahap analisis berbeda-beda tergantung literatur yang digunakan. Pada bab ini yang dimaksud analisis sistem adalah mendefinisikan kebutuhan terkait sistem yang akan dikembangkan. Jadi hasil akhir dari tahap analisis di sini adalah sebuah dokumen yang menjelaskan mengenai spesifikasi persyaratan sistem informasi atau SRS (*System Requirement Specification*)

Tujuan

1. Mahasiswa memahami mengenai kegiatan apa saja yang dilakukan selama tahap analisis sistem
2. Mahasiswa mengetahui teknik apa saja yang digunakan untuk melakukan analisis sistem
3. Mahasiswa mampu melakukan analisis sistem menggunakan teknik yang berbeda-beda
4. Mahasiswa mampu membuat sebuah dokumen spesifikasi kebutuhan sistem informasi dengan baik.

3.1 *Definisi Analisis Sistem*

Kegiatan analisis sistem adalah kegiatan untuk melihat sistem yang sudah berjalan, melihat bagian mana yang bagus dan tidak bagus, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru. Hal tersebut terlihat sederhana, namun sebenarnya tidak. Banyak hambatan yang akan ditemui dalam proses tersebut.

Pada banyak proyek sistem informasi, proses analisis dan desain sering kali berjalan bersama-sama. Jadi selama kegiatan analisis, kegiatan desain juga dilakukan. Hal ini dilakukan karena pada banyak kasus, *user* sering kesulitan untuk mendefinisikan kebutuhan mereka. Jadi mereka akan lebih mudah mendefinisikan kebutuhan, jika mereka telah melihat gambar rancangan sistem yang baru, khususnya rancangan antarmuka.

Oleh karena itu, sering kali batasan mengenai bagian mana yang dianggap sebagai analisis dan bagian mana yang dianggap sebagai desain banyak terjadi perbedaan. Misalnya ada yang mengatakan bahwa *use case*, *analysis class*, dan *sequence diagram* merupakan bagian dari analisis. Namun ada juga pihak lain yang menyatakan bahwa *use case* dan *sequence diagram* merupakan bagian dari desain, dan *analysis class* tidak ada karena sudah ada *design class*.

Pada buku ini yang dibahas pada bagian analisis adalah bagaimana metode pengumpulan data dan bagaimana mendokumentasikannya. Sedangkan *use case*, *class diagram*, dan *sequence diagram* dianggap merupakan bagian dari desain sistem dan akan dibahas pada bab yang terkait dengan UML.

3.2 *Perlunya Analisis Sistem*

Fase analisis sistem memberikan pemahaman tentang sistem yang sudah ada dan menemukan peluang untuk pengembangan sistem menjadi lebih baik serta memenuhi kebutuhan bisnis. Karena itu fase ini menjadi acuan penting dalam proyek pengembangan sistem informasi.

Pendekatan analisis sistem yang populer adalah analisis terstruktur, teknik informasi (*information engineering*), dan analisis berorientasi objek. Analisis terstruktur fokus pada aliran data melalui proses-proses bisnis dan perangkat lunak. Dikenal pula dengan nama analisis process-centered. Para analis sistem

menggambar serangkaian model proses yang disebut diagram aliran data (*data flow diagram*) yang mengilustrasikan proses-proses yang ada dan/atau yang diusulkan dalam sebuah sistem.

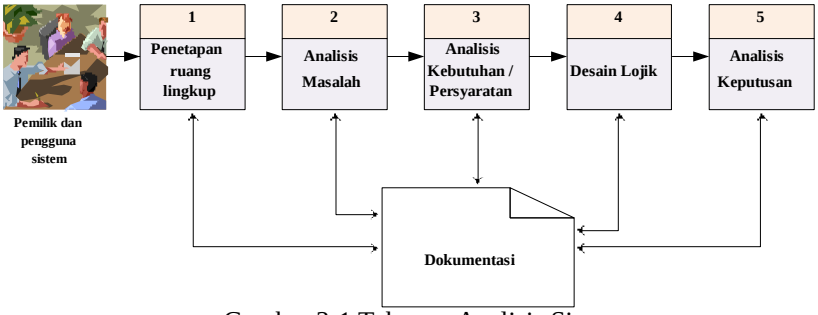
Analisis dengan teknik informasi fokus pada struktur data tersimpan dalam sebuah sistem, karena itu disebut analisis data-centered. Model-model proses dalam teknik ini digambarkan dengan diagram aliran data yang disebut hubungan entitas (*entity relationship*).

Analisis berorientasi objek menghilangkan pemisahan artifisial data dan proses, sebaliknya data dan proses yang membuat membaca memperbarui dan menghapus data itu diintegrasikan ke dalam konstruksi yang disebut objek. Unified model language (UML) adalah standar pemodelan yang menyediakan model-model objek.

3.3 *Tahapan Analisis Sistem*

Analisis sistem dikendalikan oleh kepedulian bisnis para pemilik sistem dan pengguna sistem. Para analis sistem berperan sebagai fasilitator antara pemilik dan pengguna sistem.

Tahapan analisis sistem digambarkan pada gambar 3-1 di bawah ini:



Gambar 3-1 Tahapan Analisis Sistem

Penetapan Ruang Lingkup

Fase ini memiliki tugas :

- Mengidentifikasi Masalah Awal yang ada pada sistem saat ini, seperti seberapa urgensi, tingkat visibilitas, berapa keuntungan yang akan diperoleh dari pemecahan masalah, prioritas dan penetapan solusi untuk memecahkan masalah.
- Menegosiasikan ruang lingkup untuk proyek pengembangan sistem.
- Menilai kelayakan proyek, seperti contoh di bawah ini :

Pernyataan singkat masalah atau kesempatan	Urgensi	Visibilitas	Keuntungan Tahunan	Prioritas	Solusi yang diusulkan
1. Waktu respon pesanan, diukur dari saat menerima pesanan sampai pengiriman pelanggan meningkat rata-rata 15 hari	Segera	Tinggi	$175.000	2	Pengembangan baru
2. Ketidakkonsistenan data dalam file-file anggota dan pesanan	3 bulan	Tinggi	$ 35.000	1	Perbaikan cepat, kemudian pengembangan baru

- Mengembangkan jadual dan anggaran awal.
- Mengkomunikasikan rencana proyek.

Analisis Masalah

Selalu ada sistem saat ini atau yang sudah ada, fase ini menyediakan analisis dengan pemahaman, kesempatan atau perintah lebih dalam yang memicu proyek.

Fase ini memiliki tugas :

- Memahami bidang masalah. Tim analis mencoba mempelajari sistem saat ini. Pemilik dan pengguna sistem memiliki persepsi berbeda tentang sistem yang ada, studi yang dilakukan dengan baik dapat mengungkap kepentingan semua pihak.

- Menganalisis masalah-masalah dan kesempatan-kesempatan. Meski sudah dilakukan di fase sebelumnya, tetapi masalah-masalah awal tersebut hanya gejala, bukan masalah yang dipahami oleh pengguna sistem. Analisis masalah adalah keterampilan yang sulit dikuasai, tiap masalah dianalisis penyebab dan akibatnya.
- Menganalisis proses-proses bisnis. Dikenal juga sebagai desain ulang proses bisnis. Tim analis akan memeriksa setiap proses bisnis dengan lebih rinci untuk mengukur nilai yang akan ditambahkan atau dikurangi.
- Menentukan tujuan-tujuan perbaikan sistem. Tim analis menentukan kriteria di mana semua perbaikan pada sistem akan diukur dan mengidentifikasi batasan yang membatasi fleksibilitas semua perbaikan tersebut. Kriteria sukses diukur dengan tujuan, setiap tujuan mewakili usaha. Contoh analisis sebab akibat dan penentuan tujuan perbaikan sistem adalah di bawah ini :

Analisis Sebab dan Akibat		Tujuan – tujuan Perbaikan Sistem	
Masalah atau Kesempatan	Sebab dan Akibat	Tujuan Sistem	Batasan Sistem
Waktu respon pesanan tidak dapat diterima (terlalu lama)	Sistem terlalu tergantung pada keyboard. Nilai yang sama ditujukan bagi kebanyakan pesanan.	Entri data lewat keyboard berkurang 50% untuk semua pesanan	Beberapa sistem yang dikembangkan harus cocok dengan standar desktop Windows XP Profesional SP-2

- Memperbarui rencana proyek.
- Mengkomunikasikan penemuan-penemuan dan rekomendasi.

Analisis Persyaratan

Hal fatal setelah fase analisis masalah adalah mulai melihat berbagai solusi alternatif, khususnya solusi teknis. Salah satu kesalahan yang kerap terjadi di dalam sistem informasi terbaru ditunjukkan dalam pernyataan „memastikan sistem bekerja dan secara teknis mengesankan". Ini sebaiknya mengenai „apa" dan bukan „bagaimana". Yang harus dipikirkan adalah apa yang sungguh-sungguh dibutuhkan dan diinginkan oleh pengguna dari sistem yang baru. Sistem yang baru akan selalu dievaluasi apakah memenuhi atau tidak

memenuhi sasaran dan kebutuhan bisnis, karena itu fase ini tidak dapat diabaikan.

Tugas-tugas yang terdapat pada fase ini adalah :

- Mengidentifikasi dan menyatakan kebutuhan / persyaratan bisnis. Tugas ini menerjemahkan sasaran-sasaran kedalam functional requirement. Functional requirement adalah deskripsi mengenai aktivitas dan layanan yang harus diberikan / disediakan oleh sistem.
- Membuat prioritas persyaratan sistem. Tidak semua persyaratan dibuat sama, karena tingkatan kebutuhannya berbeda, karena itu pemilik dan pengguna sistem harus membuat prioritas persyaratan.
- Memperbarui atau memperhalus rencana proyek. Ruang lingkup adalah sebuah target yang berubah. Setelah mengidentifikasi persyaratan bisnis, kita harus mundur dan menetapkan kembali pemahaman kita mengenai ruang lingkup proyek dan memperbarui rencana proyek kita untuk melakukan penyesuaian.
- Mengkomunikasikan pernyataan kebutuhan / persyaratan. Komunikasi adalah sebuah tugas fase analisis persyaratan yang berlangsung terus – menerus. Kita harus mengkomunikasikan persyaratan dan prioritas kepada komunitas bisnis melalui fase ini.

Desain Lojik

Pada fase ini kita menggambarkan berbagai model sistem untuk mendokumentasikan persyaratan untuk sistem baru dan sistem yang ditingkatkan.

Analisis Keputusan

Dengan adanya persyaratan bisnis, maka kita dapat menekankan bagaimana sistem baru dapat diimplementasikan dengan teknologi. Di fase ini kita mengenali kandidat solusi, menganalisa kandidat solusi dan merekomendasi sebuah sistem yang akan dirancang, dibangun dan diimplementasikan. Contoh analisis keputusan adalah di bawah ini :

Karakteristik	Kandidat 1	Kandidat 2	Kandidat 3	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, Internet Explorer	MS Visual Basic 7.0, System Architect 4.1, Internet Explorer	

3.4 *Jenis Kebutuhan*

Kebutuhan (*requirement*) yang dikumpulkan dengan menggunakan wawancara, observasi, kuisioner, atau gabungan dari ketiga hal tersebut dapat dikelompokkan menjadi beberapa kategori sebagai berikut (tidak semua kebutuhan ini harus ada).

- *Functional requirement*. Kebutuhan yang terkait dengan fungsi produk, misalnya sistem informasi harus mampu mencetak laporan, sistem informasi harus mampu menampilkan grafik, dan lain-lain.
- *Development requirement*. Kebutuhan yang terkait *tools* untuk pengembangan sistem informasi baik perangkat keras maupun perangkat lunak, misalnya sistem informasi dikembangkan dengan menggunakan alat bantu Eclipse untuk pengembangan dan Jude Community untuk pemodelan.
- *Deployment requirement*. Kebutuhan terkait dengan lingkungan di mana sistem informasi akan digunakan baik perangkat lunak maupun perangkat keras. Contoh kebutuhan ini misalnya sistem informasi harus mampu berjalan pada server dengan spesifikasi perangkat keras *memory* 1 GB, *processor* Pentium 4 2 GB, dan spesifikasi sistem operasi Ubuntu 7.4.
- *Performance requirement*. Kebutuhan yang terkait dengan ukuran kualitas maupun kuantitas, khususnya terkait dengan kecepatan, skalabilitas, dan kapasitas. Misalnya sistem informasi tersebut harus mampu diakses oleh minimal 1000 orang pada waktu yang bersamaan.
- *Documentation requirement*. Kebutuhan ini terkait dengan dokumen apa saja yang akan disertakan pada produk akhir. Dokumen yang biasanya dihasilkan pada tahap akhir pengembangan sistem informasi antara lain dokumen teknis (mulai dari dokumen perencanaan

proyek, analisis, desain, sampai pengujian), *user manual*, dan dokumen pelatihan.

- *Support requirement*. Kebutuhan yang terkait dukungan yang diberikan setelah sistem informasi digunakan. Dukungan teknis tersebut misalnya adanya pelatihan bagi calon pengguna.
- *Miscellaneous requirement*. Kebutuhan ini adalah kebutuhan-kebutuhan tambahan lainnya yang belum tercakup pada beberapa kategori kebutuhan yang telah terdefinisi di atas.

3.5 *Teknik Pengumpulan Data*

Hal pertama yang dilakukan dalam analisis sistem adalah melakukan pengumpulan data. Ada beberapa teknik pengumpulan data yang sering dilakukan yaitu sebagai berikut:

- Teknik Wawancara
- Teknik Observasi
- Teknik Kuisioner

Teknik Wawancara

Pengumpulan data dengan menggunakan wawancara mempunyai beberapa keuntungan sebagai berikut:

- Lebih mudah dalam menggali bagian sistem mana yang dianggap baik dan bagian mana yang dianggap kurang baik
- Jika ada bagian tertentu yang menurut anda perlu untuk digali lebih dalam, anda dapat langsung menanyakan kepada narasumber
- Dapat menggali kebutuhan *user* secara lebih bebas
- *User* dapat mengungkapkan kebutuhannya secara lebih bebas.

Selain mempunyai beberapa kelebihan tersebut, teknik wawancara juga mempunyai beberapa kelemahan. Berikut ini adalah beberapa kelemahan dari teknik wawancara:

- Wawancara akan sulit dilakukan jika narasumber kurang dapat mengungkapkan kebutuhannya
- Pertanyaan dapat menjadi tidak terarah, terlalu fokus pada hal-hal tertentu dan mengabaikan bagian lainnya.

Berikut ini adalah beberapa panduan dalam melakukan kegiatan wawancara agar memperoleh data yang diharapkan:

- Buatlah jadwal wawancara dengan narasumber dan beritahukan maksud dan tujuan wawancara
- Buatlah panduan wawancara yang akan anda jadikan arahan agar pertanyaan dapat fokus kepada hal-hal yang dibutuhkan. Panduan wawancara antara lain adalah :

Yang Harus Dilakukan	Yang Harus Dihindari
▪ Bersikap sopan ▪ Jadilah pendengar yang baik ▪ Terkendali ▪ Menyelidiki ▪ Amati perangainya dan komunikasi nonverbalnya ▪ Sabar ▪ Menjaga sikap formal tapi santai	▪ Melontarkan pertanyaan yang tidak perlu ▪ Lebih banyak berbicara dibanding mendengarkan ▪ Menggunakan kata-kata jargon dan kasar ▪ Berdebat dengan partisipan ▪ Mengkritik dan menyindir partisipan

- Gunakan pertanyaan yang jelas dan mudah dipahami. Hindari pertanyaan yang panjang dan kompleks.
- Wawancara umumnya terdiri dari tiga fase yaitu pembukaan, isi dan kesimpulan. Pembukaan bertujuan mempengaruhi atau memotivasi orang yang diwawancarai (narasumber) untuk berpartisipasi dan berkomunikasi dengan membangun lingkungan/suasana yang ideal. Isi adalah fase dimana pewawancara memberikan pertanyaan kemudian mendengarkan/mengamati dengan baik jawaban verbal maupun nonverbal dari partisipan. Kesimpulan merupakan tahap akhir dimana pewawancara menunjukkan penghargaan dan menyampaikan kesimpulan dari hasil wawancara.
- Cobalah untuk menggali mengenai kelebihan dan kekurangan sistem yang telah berjalan sebelumnya.
- Anda boleh berimprovisasi dengan mencoba menggali bagian-bagian tertentu yang menurut anda penting, misalnya melewati pertanyaan-pertanyaan yang sudah dijawab di pertanyaan sebelumnya, atau dapat dihapus jika dianggap tidak relevan berdasarkan informasi yang sudah diketahui secara pasti selama wawancara.
- Catat hasil wawancara tersebut.

3.1.1 Teknik Observasi

Pengumpulan data dengan menggunakan observasi mempunyai keuntungan yaitu :

- Analis dapat melihat langsung bagaimana sistem lama berjalan
- Mampu menghasilkan gambaran lebih baik jika dibanding dengan teknik lainnya.

Sedangkan kelemahan dengan menggunakan teknik observasi adalah :

- Membutuhkan waktu cukup lama karena jika observasi waktunya sangat terbatas maka gambaran sistem secara keseluruhan akan sulit untuk diperoleh
- Orang-orang yang sedang diamati biasanya perilakunya akan berbeda dengan perilaku sehari-hari (cenderung berusaha terlihat baik). Hal ini akan menyebabkan gambaran yang diperoleh selama observasi akan berbeda dengan perilaku sehari-hari
- Dapat mengganggu pekerjaan orang-orang pada bagian yang sedang diamati.

Berikut ini adalah beberapa petunjuk untuk melakukan observasi :

- Tentukan hal-hal apa saja yang akan diobservasi agar kegiatan observasi menghasilkan sesuai dengan yang diharapkan
- Mintalah ijin kepada orang yang berwenang pada bagian yang akan diobservasi
- Berusaha sesedikit mungkin agar tidak menganggu pekerjaan orang lain
- Jika ada yang anda tidak mengerti, cobalah bertanya. Jangan membuat asumsi sendiri.

3.1.2 Teknik Kuisioner

Pengumpulan data dengan menggunakan kuisioner mempunyai keuntungan yaitu :

- Hasilnya lebih objektif, karena kuisioner dapat dilakukan kepada banyak orang sekaligus
- Waktunya lebih singkat.

Sedangkan kelemahan pengumpulan data dengan menggunakan kuisioner adalah sebagai berikut :

- Responden cenderung malas untuk mengisi kuisioner
- Sulit untuk membuat pertanyaan yang singkat, jelas, dan mudah dipahami.

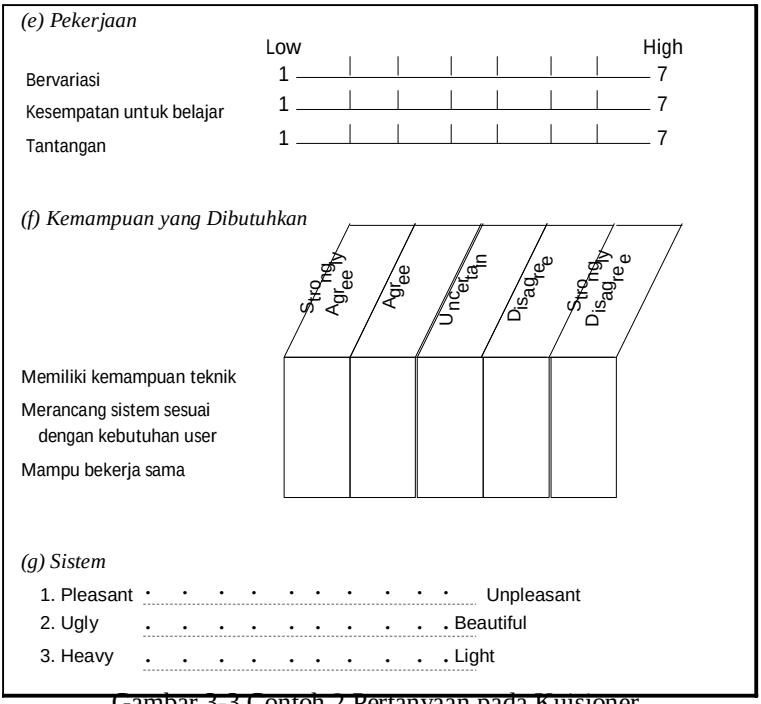
Berikut ini adalah beberapa cara yang dapat dilakukan untuk membuat teknik kuisioner menghasilkan data yang baik :

- Hindari pertanyaan isian, karena responden biasanya malas untuk menulis banyak, dan jika responden menuliskan sesuatu sering kali susah untuk dipahami. Contoh pertanyaan yang memudahkan responden adalah pilihan ganda. Pertanyaan pilihan ganda memudahkan anda untuk melakukan rekapitulasi data hasil kuisoner
- Buatlah pertanyaan yang tidak terlalu banyak
- Buatlah pertanyaan yang singkat, padat, dan jelas.

Di bawah ini adalah contoh-contoh pertanyaan di dalam kuisioner :

Gambar 3-2 Contoh Pertanyaan pada Kuisioner

Pertanyaan (a) adalah pilihan berganda, responden tinggal memberi silang pada jawaban yang dianggap tepat. Pertanyaan (b) memberikan tanda cek (v) pada kolom jawaban di sebelah kanan. Pertanyaan (c) responden memberikan tanda cek atau keterangan Y atau T untuk garis isian. Pertanyaan (d) responden diharuskan mengisi, contoh „Centrino Duo" pada pertanyaan Intel, dan seterusnya.

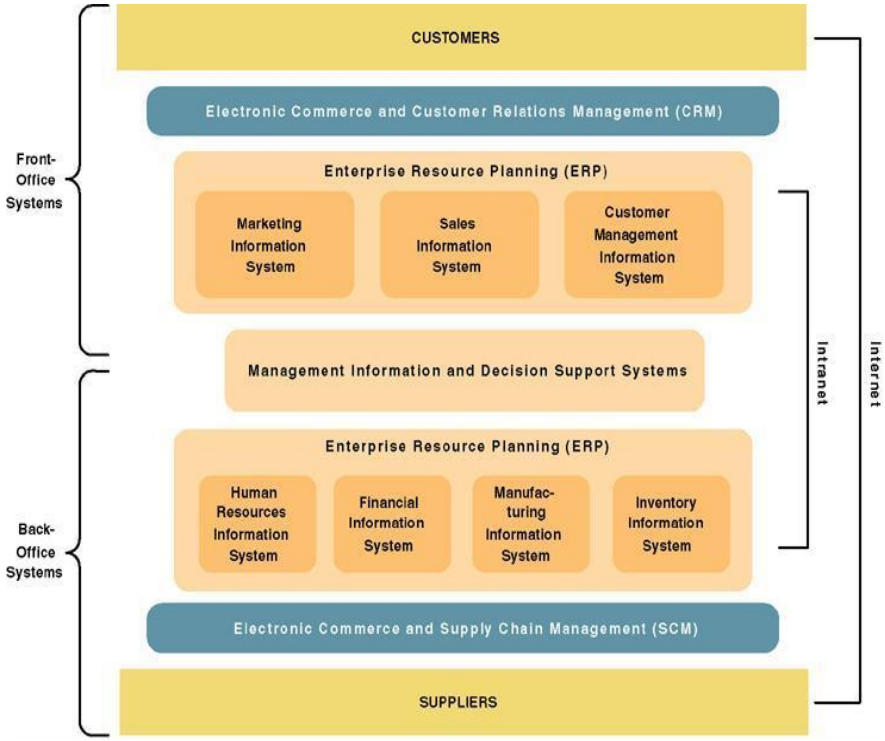


Gambar 3-3 Contoh 2 Pertanyaan pada Kuisioner

Pertanyaan (e) dan (g) responden menjawab dengan cara mengarsir bagian kiri atau kanan, semakin luas daerah yang diarsir maka semakin tinggi penekanan jawaban responden. Pertanyaan (f) responden menjawab dengan cara memberi tanda silang (x) pada kolom jawaban yang dianggap benar.

3.6 *Blok Pembangun Sistem Informasi*

Organisasi tidak hanya dilayani oleh satu sistem informasi, melainkan oleh beberapa sistem informasi yang masing-masing mendukung fungsi bisnis tertentu, dapat dilihat pada gambar di bawah ini.



Gambar 3-4 Kesatuan Sistem Informasi yang Mendukung Organisasi
Sumber : Whitten, System Analysis and Design Method, 2004

Sistem informasi *front-office* mendukung fungsi bisnis yang mencapai konsumen, sedangkan sistem informasi *back-office* mendukung operasi bisnis internal dan berinteraksi dengan pemasok. Sistem informasi *front* dan *back office* mengalirkan data ke sistem informasi manajemen dan sistem pendukung keputusan yang menyokong kebutuhan bisnis manajemen. Sistem informasi

berinteraksi dengan pelanggan dan pemasok menggunakan aplikasi teknologi perdagangan elektronik (e-commerce), manajemen hubungan pelanggan (CRM), manajemen rantai persediaan (SCM) di internet.

Arsitektur sistem informasi berperan sebagai kerangka tingkat tinggi untuk memahami pandangan-pandangan yang berbeda mengenai **blok pembangun dasar** sebuah sistem informasi. Pandangan yang berbeda dikarenakan setiap komponen sistem melihat dari sisinya masing-masing, misalnya pengguna sistem fokus pada tujuan bisnis secara umum, para desainer fokus pada teknologi yang mungkin dapat digunakan sistem informasi untuk meraih tujuan bisnis, dan seterusnya. Gambar 3-5 menjelaskan mengenai blok pembangun dasar sistem informasi.

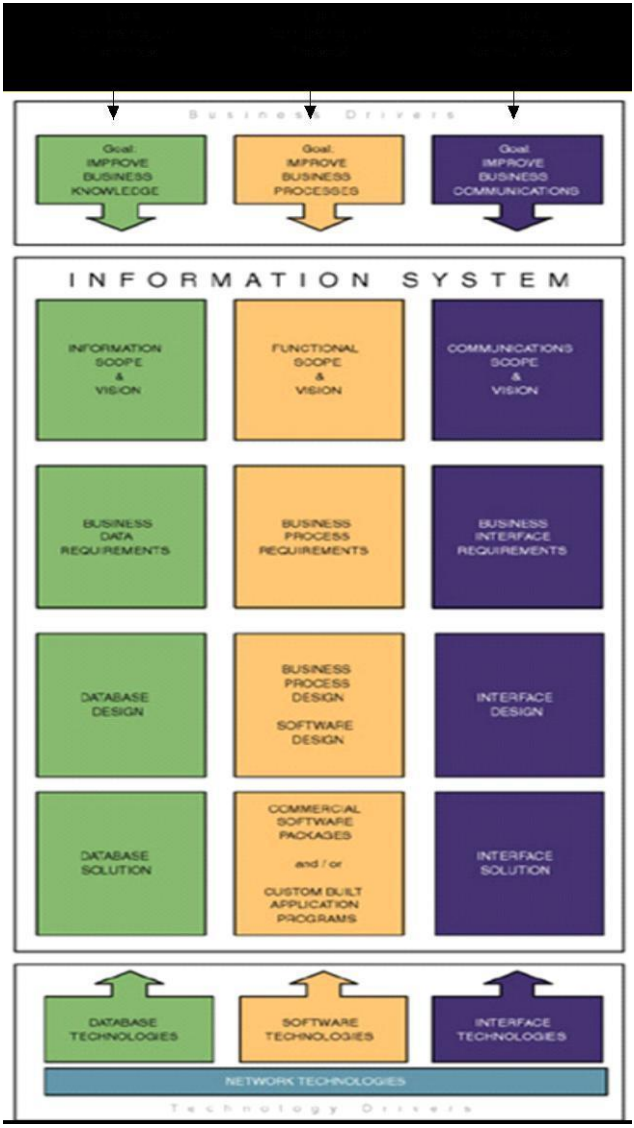
Blok-blok pembangun sistem informasi adalah (a) blok pembangun pengetahuan, (b) blok pembangun proses dan (c) blok pembangun komunikasi.

Blok Pembangun Pengetahuan, dapat dilihat pada gambar 3-5 sisi sebelah kiri, bertujuan mendapatkan dan menyimpan data bisnis dengan menggunakan teknologi basis data (seperti Access, SQL Server, Oracle). Setiap stakeholder memiliki pandangan berbeda mengenai **pengetahuan**. Pemilik sistem tidak tertarik pada data mentah melainkan pada informasi yang menambahkan pengetahuan bisnis baru dan membantu manajer mengambil keputusan cerdas yang sesuai dengan misi, tujuan, sasaran dan sisi kompetitif organisasi. Pengguna sistem memandang pengetahuan sebagai data yang dapat disimpan dalam dua bentuk, yaitu dalam kabinet file atau disimpan dalam file (basis data) komputer. Desainer sistem memandang pengetahuan sebagai struktur data, skema basis data, field, index dan *contraint* basis data. Sedangkan pembangun sistem memandang pengetahuan sebagai bahasa SQL dan teknologi DBMS.

Blok Pembangun Proses, dapat dilihat pada gambar 3-5 di bagian tengah, mewakili kerja dalam sistem. Pada bagian bawah kolom proses adalah teknologi perangkat lunak yang dapat digunakan untuk mengotomatisasi proses-proses yang sudah ditentukan. Mari kita bahas pandangan para stakeholder mengenai proses. Pemilik sistem tertarik pada kelompok proses tingkat tinggi yang disebut fungsi bisnis. Fungsi bisnis adalah sekelompok proses yang berkaitan yang menyokong bisnis. Pengguna sistem tertarik pada pekerjaan yang harus dilakukan untuk menyediakan respon yang sesuai dengan kejadian bisnis, misalnya proses bisnis, process requirement,

kebijakan, prosedur dan aliran kerja (work flow). Proses bisnis adalah kegiatan yang merespon kejadian bisnis sedangkan process requirement adalah harapan pengguna terhadap sebuah proses bisnis dan sistem informasi. Desainer sistem tertarik pada proses-proses apa yang dapat diotomatisasikan dan bagaimana caranya. Sedangkan pembangun sistem tertarik pada logika program yang akan mengimplementasikan otomatisasi proses, misalnya bahasa program aplikasi apa yang akan digunakan dan seterusnya.

Blok Pembangun Komunikasi, dapat dilihat pada gambar 3-5 sisi sebelah kanan, dimana di bawah kolom komunikasi adalah teknologi antarmuka untuk mengimplementasikan antarmuka komunikasi. Tujuan umum organisasi adalah memperbaiki komunikasi dan kolaborasi bisnis, perbaikan komunikasi umumnya diarahkan ke tujuan antarmuka yaitu menyediakan antarmuka yang efektif dan efisien bagi pengguna sistem. Mari kita lihat pandangan para stakeholder terhadap komunikasi. Pemilik tertarik pada siapa yang akan berinteraksi dengan sistem. Pengguna tertarik pada input dan/atau output sistem informasi. Desainer sistem tertarik pada desain teknik antarmuka antar sistem. Sedangkan pembangun sistem tertarik pada pembuatan, instalasi, pengujian dan implementasi antarmuka.



Gambar 3-5 Blok Pembangun Sistem Informasi

3.7 *Dokumen Spesifikasi Kebutuhan Sistem*

Analis sistem menggunakan berbagai alat untuk mendokumentaskan penemuan mmereka pada saat menganalisis sistem. Dokumen tersebut dikenal sebagai dokumen spesifikasi kebutuhan sistem (System Requirement Specification / SRS).

1. PENDAHULUAN
   - 1.1 Tujuan
   - 1.2 Ruang Lingkup
   - 1.3 Definisi
   - 1.4 Referensi
   - 1.5 Sistematika
2. DESKRIPSI UMUM
   - 2.1 Perspektif
   - 2.2 Kegunaan
   - 2.3 Karakteristik Pengguna
   - 2.4 Batasan-batasan
   - 2.5 Asumsi dan Ketergantungan
3. SPESIFIKASI KEBUTUHAN
   - 3.1 Kebutuhan Fungsional
     - 3.1.1 Pendahuluan
     - 3.1.2 Input
     - 3.1.3 Proses
     - 3.1.4 Output
   - 3.2 Kebutuhan Antarmuka Eksternal
     - 3.2.1 Antarmuka Pengguna
     - 3.2.2 Antarmuka Perangkat Keras
     - 3.2.3 Antarmuka Perangkat Lunak
     - 3.2.4 Antarmuka Komunikasi
   - 3.3 Kebutuhan Performansi
   - 3.4 Kendala Disain
     - 3.4.1 Standard Compliance
     - 3.4.2 Perangkat Keras
   - 3.5 Atribut
     - 3.5.1 Keamanan Sistem
     - 3.5.2 Pemeliharaan
   - 3.6 Kebutuhan Lain
     - 3.6.1 Database
     - 3.6.2 Pengoperasian
     - 3.6.3 Penyesuaian Tempat

Soal Benar Salah

1. Kegiatan analisis sistem adalah kegiatan untuk melihat sistem yang sudah berjalan, untuk melihat bagian mana yang bagus dan bagian mana yang seharusnya diperbaiki.
2. Fase analisis sistem memberikan pemahaman tentang sistem yang sudah ada dan menemukan peluang untuk pengembangan sistem menjadi lebih baik serta memenuhi kebutuhan bisnis. Karena itu fase ini boleh diabaikan.
3. Tahapan analisis sistem adalah Penetapan ruang lingkup, Analisis Masalah, Desain Lojik dan Analisis Keputusan.
4. Teknik pengumpulan data dapat dilakukan dengan cara wawancara, observasi di lapangan dan penyebaran angket.
5. Sebelum melakukan wawancara sebaiknya membuat panduan sebagai arahan agar pertanyaan dapat fokus kepada hal-hal yang dibutuhkan.
6. Agar hasil kuisioner lebih objektif, maka bentuk pertanyaan harus dibuat berupa isian dan essay.
7. Blok pembangun sistem informasi adalah sebuah arsitektur berperan sebagai kerangka tingkat tinggi untuk memahami pandangan-pandangan yang berbeda sebuah sistem informasi.
8. Pada blok Pembangun Pengetahuan, pengguna sistem memandang pengetahuan sebagai informasi berharga yang membantu mereka dalam mengambil keputusan.
9. Functional requirement adalah jenis kebutuhan yang terkait dengan fungsi produk.
10. Dokumen yang digunakan analis sistem untuk merekam temuan mereka selama fase analisis sistem disebut dokumen spesifikasi persyaratan sistem.

Soal Pilihan Ganda

1. Di bawah ini adalah pengertian analisis sistem, yaitu :
   a. Kegiatan untuk melihat sistem yang sudah berjalan, melihat bagian mana yang bagus dan tidak bagus, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru.
   b. Kegiatan untuk melihat kondisi proyek sistem informasi di tahap desain sistem, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru.
   c. Kegiatan untuk melihat sistem yang sudah berjalan, melihat departemen di organisasi tersebut untuk dilakukan rekruting, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru.
   d. Kegiatan untuk melihat kondisi bisnis suatu organisasi, dan kemudian melakukan kegiatan jual beli untuk memenuhi sistem yang baru.
   e. Salah semua.

2. Tahapan analisis sistem adalah di bawah ini, kecuali :
   a. Penetapan anggota tim proyek
   b. Analisis masalah
   c. Analisis kebutuhan sistem
   d. Desain lojik
   e. Analisis keputusan

3. Membuat prioritas persyaratan sistem adalah termasuk tugas yang terdapat pada tahap :
   a. Penetapan anggota tim proyek
   b. Analisis masalah
   c. Analisis kebutuhan sistem
   d. Desain lojik
   e. Analisis keputusan

4. Hal-hal yang sebaiknya dihindari pada saat wawancara adalah :
   a. Melontarkan pertanyaan yang diperlukan
   b. Lebih banyak mendengarkan
   c. Gunakan kata-kata yang baik, jelas dan sopan
   d. Sabar
   e. Menasihati partisipan

5. Salah satu keuntungan teknik observasi pada saat mengumpulkan data adalah :
   a. Waktu yang dibutuhkan lama sehingga memudahkan pekerjaan
   b. Perilaku orang yang diamati berbeda-beda dari keadaan yang sebenarnya
   c. Tidak mengganggu orang yang sedang diamati
   d. Dapat melihat langsung bagaimana sistem lama berjalan
   e. Merupakan teknik pengumpulan data yang paling mudah

6. Blok pembangun sistem informasi antara lain adalah blok pembangun proses. Pernyataan yang salah di bawah ini berkaitan dengan blok tersebut adalah :
   a. Pemilik sistem memandang proses sebagai fungsi bisnis
   b. Pengguna sistem memandang proses sebagai kejadian bisnis atau proses bisnis
   c. Desainer sistem memandang proses sebagai cara untuk mengotomatisasikan kegiatan di dalam organisasi
   d. Pembangun sistem memandang proses sebagai logika program
   e. Salah semua

7. Blok pembangun komunikasi bertujuan untuk :
   a. Memperbaiki komunikasi dengan pemerintah
   b. Memperbaiki hubungan dengan pelanggan dan suplier
   c. Memperbaiki jaringan dan teknologi informasi
   d. Memperbaiki antarmuka bagi pengguna sistem
   e. Memperbaiki komunikasi pada sistem terpusat

8. Pernyataan yang benar di bawah ini berkaitan dengan blok pembangun komunikasi adalah :
   a. Dibangun untuk meningkatkan citra di mata pelanggan.
   b. Pemilik sistem tertarik pada siapa saja yang berinteraksi dengan sistem.
   c. Pengguna sistem tertarik pada laporan bulanan.
   d. Desainer sistem tertarik pada siapa saja para staf dan manajemen yang memegang komputer.
   e. Pembangun sistem tertarik pada meningkatkan kemampuan komunikasi verbal di dalam organisasi.

9. Pada blok pembangun pengetahuan, pandangan para stakeholder mengenai pengetahuan adalah di bawah ini, kecuali :
   a. Pemilik sistem tertarik pada data mentah
   b. Pemilik sistem tidak tertarik pada data mentah
   c. Pengguna sistem memandang pengetahuan sebagai data
   d. Desainer sistem memandang pengetahuan sebagai struktur data
   e. Pembangun sistem memandang pengetahuan sebagai DBMS

10. Kebutuhan yang terkait tools untuk pengembangan sistem informasi baik perangkat keras maupun perangkat lunaknya, misalnya software Jude Community. Kebutuhan tersebut adalah jenis kebutuhan :
    a. Functional Requirement
    b. Development Requirement
    c. Deployment Requirement
    d. Performance Requirement
    e. Documentation Requirement

PDF-XChange
Click to buy NOW!
www.docu-track.com

4 Desain Sistem

Overview

Desain sistem informasi merupakan tahapan yang harus dilakukan berikutnya setelah analisis desain. Bab desain sistem ini berisi konsep dasar untuk melakukan perancangan sistem dengan menggunakan pendekatan berorientasi objek. Selain itu, pada bab ini juga ada perbandingan metode desain sistem berorientasi objek dengan pendekatan terstruktur.

Tujuan

1. Mahasiswa memahami apa yang dilakukan dalam tahap desain sistem.
2. Mahasiswa memahami desain sistem dengan pendekatan berorientasi objek.
3. Mahasiswa mampu membuat desain sistem dengan pendekatan berorientasi objek.
4. Mahasiswa mengerti perbandingan antara pendekatan berorientasi objek dengan pendekatan terstruktur

4.1 *Definisi Desain Sistem*

Desain atau perancangan dalam pembangunan perangkat lunak merupakan upaya untuk mengonstruksi sebuah sistem yang memberikan kepuasan (mungkin informal) akan spesifikasi kebutuhan fungsional, memenuhi target, memenuhi kebutuhan secara implisit atau eksplisit dari segi performansi maupun penggunaan sumber daya, kepuasan batasan pada proses desain dari segi biaya, waktu, dan perangkat. Kualitas perangkat lunak biasanya dinilai dari segi kepuasan pengguna perangkat lunak terhadap perangkat lunak yang digunakan.

4.2 *Konsep Dasar Pendekatan Berorientasi Objek*

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak. sistem informasi, atau sistem lainnva). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nvata. Ada banvak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek mi, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennva dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya. dan dapat berinteraksi satu sama lain.

Karakteristik atau sifat-sifat yang dipunyai sebuah sistem berorientasi objek adalah sebagai berikut:

- Abstraksi
  prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan
- Enkapsulasi
  pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek. untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya
- Pewarisan (*inheritance*)
  mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya
- *Reusabilily*
  pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut
- Generalisasi dan Spesialisasi
  menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus
- Komunikasi Antar Objek
  komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya
- *Polymorphism*
  kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

4.3 *Metodologi Berorientasi Objek*

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnva. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi

objek. perancangan berorientasi objek, pemrograman berorientasi objek. dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis. *real-time, utility*, dan sebagainva) dengan platform yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

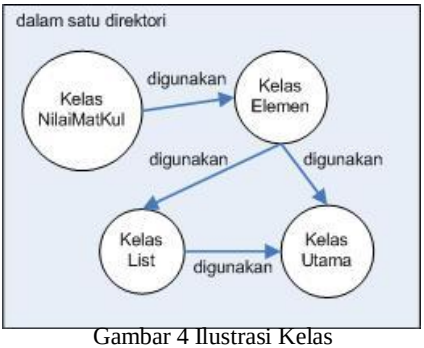
Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

- meningkatkan produktivitas
  karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnva yang melibatkan objek tersebut (*reusable*)
- kecepatan pengembangan
  karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnva kesalahan pada saat pengkodean
- kemudahan pemeliharaan
  karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah
- adanya konsistensi
  karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
- meningkatkan kualitas perangkat lunak
  karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannva, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

4.4 *Pengertian Objek dan Kelas*

Kelas adalah kumpulan dari objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek. Ilustrasi dari sebuah kelas dapat dilihat pada gambar berikut.



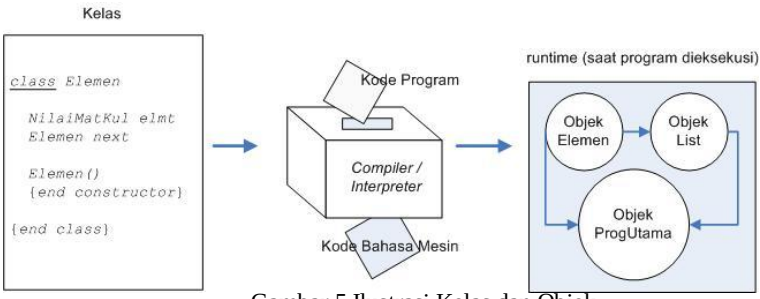
Gambar 4 Ilustrasi Kelas

Sebuah kelas lebih fleksibel untuk digunakan oleh kelas lain.

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunvai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat dari segi teknis adalah elemen pada saat *runtime* yang

akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi, jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas. Ilustrasi kelas dan objek dapat dilihat pada gambar berikut.



Gambar 5 Ilustrasi Kelas dan Objek

4.5 *Enkapsulasi*

Enkapsulasi dapat dianggap sebagai sebuah bungkusan. Enkapsulasi inilah yang diimplementasikan dalam sebuah kelas dimana di dalam sebuah kelas terdiri dari atribut dan metode yang dibungkus dalam suatu kelas. Enkapsulasi pada sebuah kelas bertujuan untuk melindungi atribut dan metode-metode yang ada di dalam kelas agar tidak sembarangan diakses oleh kelas lain.

4.6 *Atribut*

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

4.7 *Operasi atau Metode (*Method*)*

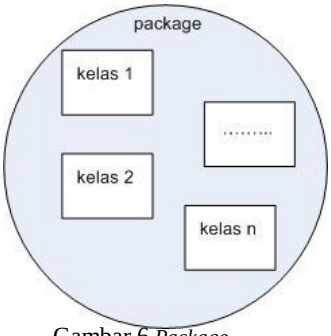
Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari

- event
- aktivitas atau aksi keadaan
- fungsi
- kelakuan dunia nyata

Contoh metode atau operasi misalnya Read, Write, Move. Copy. dan sebagainya.

4.8 *Pengertian* Package

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda. Ilustrasi dari sebuah *package* dapat dilihat pada gambar berikut.



Gambar 6 *Package*

4.9 *Pengertian Antarmuka (*Interface*)*

Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka, misalnya pada antarmuka terdapat deklarasi metode printAnimal() maka pada kelas yang mengimplementasikan metode itu harus ditulis sama. Antarmuka atau *interface* biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

4.10 *Sekilas Pendekatan terstruktur*

Teknik terstruktur, merupakan pendekatan formal untuk memecahkan masalah-masalah dalam aktivitas bisnis menjadi bagian-bagian kecil yang dapat diatur dan berhubungan untuk kemudian dapat disatukan kembali menjadi satu kesatuan yang dapat dipergunakan untuk memecahkan masalah.

Dalam hubungannya dengan pengembangan sistem informasi dan *software* aplikasi sistem informasi, pemrograman terstruktur adalah proses yang berorientasi kepada teknik yang digunakan untuk merancang dan menulis program secara jelas dan konsisten. Desain terstruktur merupakan salah satu proses yang berorientasi teknik yang digunakan untuk memilah-milah program besar ke dalam hirarki modul-modul yang menghasilkan program komputer yang lebih kecil agar mudah untuk diimplementasikan dan dipelihara (diubah). Analisis Terstruktur Modern merupakan teknik yang berorientasi kepada proses yang paling populer dan banyak digunakan dewasa ini. Pemodelan data merupakan suatu teknik yang berorientasi kepada data dengan menunjukkan sistem hanya datanya saja terlepas dari bagaimana data tersebut akan diproses atau digunakan untuk menghasilkan informasi. Rekayasa Informasi merupakan perpaduan dari pemodelan data dan proses, juga memberikan penekanan baru terhadap pentingnya perencanaan sistem informasi.

Ciri-ciri utama teknik terstruktur adalah sebagai berikut:

- merancang berdasar modul
  modularisasi adalah proses yang membagi suatu sistem menjadi beberapa modul yang dapat beroperasi secara independen
- bekerja dengan pendekatan *top-down*
  dimulai dari level atas (secara global) kemudian diuraikan sampai ke tingkat modul (rinci)
- dilakukan secara iterasi
  dengan iterasi akan didapat hasil yang lebih baik, terlalu banyak iterasi juga akan menurunkan hasilnya dan menunjukkan bahwa tahap sebelumnya tidak dilakukan dengan baik
- kegiatan dilakukan secara paralel
  pengembangan subsistem-subsistem dapat dilakukan secara paralel, sehingga akan memperpendek waktu pengembangan sistem

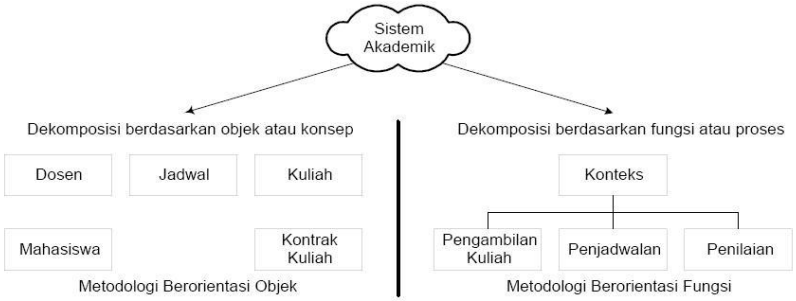
Secara teknis berikut adalah gambar ilustrasi teknik terstruktur di dalam program:

Gambar 7 Ilustrasi Teknik Terstruktur

4.11 *Perbandingan Pendekatan OO dan Terstruktur*

Perbedaan yang paling dasar dari pendekatan terstruktur dan pendekatan OO (*Object Oriented*) atau berorientasi objek adalah pada metode berorientasi fungsi atau aliran data (*Data Flow Diagram* (DFD)) (pendekatan terstruktur), dekomposisi permasalahan dilakukan berdasarkan fungsi atau proses secara hirarki, mulai dan konteks sampai proses-proses yang paling kecil, sementara pada metode berorientasi objek. dekomposisi permasalahan dilakukan berdasarkan objek-objek yang ada dalam sistem. Ilustrasi perbandingan

pendekatan berorientasi objek dengan pendekatan terstruktur dapat dilihat pada gambar berikut:



Gambar 8 Ilustrasi Perbandingan OO vs Terstruktur

Soal Benar Salah

1. Pengertian desain sistem adalah merancang sebuah sistem yang sesuai kebutuhan pengguna sistem.
2. Pendekatan berorientasi sistem adalah memandang sistem sebagai suatu objek hidup.
3. Objek adalah abstraksi sesuatu yang mewakili dunia internasional seperti benda, manusia, tempat, dan lain-lain.
4. Kelas adalah kumpulan dari tabel dengan karakteristik yang sama.
5. Sebuah kelas akan mempunvai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti.
6. Salah satu karakteristik sistem berbasis objek adalah Enkapsulasi. Enkapsulasi adalah pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek.
7. Salah satu keuntungan menggunakan metodologi berorientasi objek adalah kecepatan pengembangan karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan menyebabkan berkurangnya kesalahan pada saat pengkodean.
8. Pengembangan sistem berbasis objek meningkatkan kualitas perangkat lunak karena pengembangan jauh dari dunia nyata.
9. Ciri utama metodologi terstruktur adalah merancang sistem berdasarkan modul (modularitas).
10. Pemodelan dengan metodologi terstruktur menggunakan diagram aliran data (Data Flow Diagram / DFD), pemodelan dengan metodologi berorientasi objek menggunakan diagram hubungan entitas (E-R Diagram).

Latihan

1. *Basic*
    a. Apakah yang dimaksud dengan desain sistem?
    b. Hal apa saja yang dilakukan pada tahap desain sistem?
    c. Sebutkan dan jelaskan karakteristik apa saja yang terdapat pada sistem berorientasi objek!
    d. Mengapa berkembang metodologi berorientasi objek?
    e. Apa yang dimaksud dengan kelas dan objek? Gambarkan keterhubungan antara kelas dan objek!
    f. Apa yang dimaksud dengan pendekatan terstruktur?
    g. Apa perbedaan pendekatan terstruktur dengan pendekatan berorientasi objek?
2. *Advanced*
    a. Sebutkan minimal 2 metodologi lain (selain pendekatan terstruktur dan pendekatan berorientasi objek).
    b. Sebutkan karakteristiknya dan kemudian carilah kelebihan dan kekurangannya jika dibandingkan dengan pendekatan terstruktur dan pendekatan berorientasi objek.

5 Pengenalan UML dan Analisi *Use Case*

Overview

Bab ini berisi penjelasan mengenai pemodelan dan UML. Pemodelan digunakan untuk menggambarkan desain sistem. Salah satu bentuk pemodelan adalah UML. UML terdiri dari bermacam-macam diagram. Pada saat melakukan desain sistem, tidak harus semua diagram pada UML diimplementasikan.

Tujuan

1. Mahasiswa mampu memahami fungsi pemodelan yang digunakan untuk desain sistem.
2. Mahasiswa mengenal apa itu UML.
3. Mahasiswa mampu memahami fungsi UML
4. Mahasiswa memahami fungsi *use case*.
5. Mahasiswa mampu melakukan perancangan sistem informasi dengan menggunakan *use case*.

5.1 Kompleksitas Pengembangan Perangkat Lunak

Mengelola pengembangan perangkat lunak bukanlah hal yang mudah. Secara logika sama dengan mengelola banyak kepala yang memiliki tingkat pemahaman dan pemikiran yang berbeda untuk membuat sebuah benda. Semakin banyak kepala yang harus disatukan maka semakin sulit mengelolanya.

Kompleksitas sebuah perangkat lunak dapat dilihat dari hal-hal berikut:

- Kompleksitas domain atau permasalahan perangkat lunak
  Pendefinisian fungsi-fungsi pada perangkat dan pendefinisian penanganan kasus-kasus yang mungkin di dalam sebuah fungsi bukanlah hal yang mudah. Sering permasalahan yang belum didefinisikan pada spesifikasi muncul begitu sebuah perangkat lunak sudah masuk ke tahap implementasi atau pengkodean. Hal seperti ini dapat menyebabkan proses kembali ke tahap analisis atau sering langsung diputuskan pada saat implementasi tanpa memperbaiki dokumen analisis perangkat lunak, secara konsep hal ini dapat menyebabkan ketidakkonsistenan antara dokumen dan perangkat lunak. Belum lagi permasalahan perbedaan interpretasi pemahaman spesifikasi oleh orang yang mengembangkan aplikasi yang terkadang tidak ditanyakan, tapi membuat asumsi sendiri.
- Kesulitan mengelola proses pengembangan perangkat lunak
  Pengembangan perangkat lunak biasa dilakukan secara tim. Oleh karena itu diperlukan koordinasi yang cukup tinggi diantara anggota tim. Jika koordinasi kurang maka kesalahpahaman interpretasi akan banyak terjadi sehingga bisa jadi perangkat lunak yang dibangun tidak pernah selesai.
- Kemungkinan fleksibilitas perubahan perangkat lunak
  Sering kasus pada dunia nyata bahwa tim yang mengerjakan bukanlah orang yang memiliki kepentingan terhadap aplikasi. Sehingga tercipta hubungan klien dan developer dimana klien akan melakukan permintaan aplikasi dengan fungsi-fungsi yang dia butuhkan dan developer berkewajiban membuat aplikasi yang diminta oleh klien. Semakin terbatasnya pengetahuan klien mengenai teknologi informasi dan semakin banyaknya klien yang perlu dimintai pertimbangan mengenai kebutuhan aplikasi maka akan semakin tinggi kemungkinan perubahan spesifikasi di tengah proses pengembangan jika tidak ada perjanjian spesifikasi yang mengikat. Hal inilah yang

sering menyebabkan pengembangan aplikasi mengalami kemoloran waktu.

- Permasalahan karakteristik bagian-bagian perangkat lunak secara diskrit
  Dalam pembangunan perangkat lunak sering dilakukan dengan beberapa orang atau tim. Maka perangkat lunak akan dibagi-bagi menjadi bagian-bagian yang harus dikerjakan orang-orang di dalam tim. Jika permasalahan bagian-bagian perangkat lunak secara diskrit tidak terdefinisi dengan benar atau dipahami berbeda oleh setiap orang yang ada di dalam tim, maka kemungkinan saat bagian-bagian ini digabungkan akan terjadi banyak kesalahan atau *error*. Maka dari itu koordinasi dan komunikasi yang baik di dalam sebuah tim sangat dibutuhkan.

Karena berbagai masalah dan resiko yang mungkin timbul di dalam pengembangan perangkat lunak maka perlu adanya perencanaan dan pemodelan perangkat lunak.

5.2 Pemodelan

Pemodelan adalah gambaran dari realita yang simpel dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas misalnya jika seorang arsitek ingin memodelkan sebuah gedung yang akan dibangun maka dia akan memodelkannya dengan membuat sebuah maket (tiruan) arsitektur gedung yang akan dibangun dimana maket itu akan dibuat semirip mungkin dengan desain gedung yang akan dibangun agar arsitektur gedung yang diinginkan dapat terlihat. Seperti yang kita ketahui bahwa manusia akan lebih memahami suatu hal dengan menggunakan visual agar sekelompok manusia yang berkepentingan dapat mengerti bagaimanakah ide yang akan dikerjakan. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang yang mungkin terjadi dapat diminimalisir.

Pada dunia pembangunan perangkat lunak sistem informasi juga diperlukan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem informasi sehingga lebih terencana. Seperti halnya maket, pemodelan pada pembangunan perangkat lunak digunakan untuk memvisualkan perangkat lunak yang akan dibuat.

Pemodelan perangkat lunak memiliki beberapa abstraksi, misalnya sebagai berikut:

- petunjuk yang terfokus pada proses yang dimiliki oleh sistem
- spesifikasi struktur secara abstrak dari sebuah sistem (belum detail)
- spesifikasi lengkap dari sebuah sistem yang sudah final
- spesifikasi umum atau khusus sistem
- bagian penuh atau parsial dari sebuah sistem

Perangkat pemodelan adalah suatu model yang digunakan untuk menguraikan sistem menjadi bagian-bagian yang dapat diatur dan mengkomunikasikan ciri konseptual dan fungsional kepada pengamat. Peran perangkat pemodelan:

- Komunikasi
  - Perangkat pemodelan dapat digunakan sebagai alat komunikasi antara pemakai dengan analis sistem maupun *developer* dalam pengembangan sistem.
- Eksperimentasi
  - Pengembangan sistem yang bersifat "*trial and error*"
- Prediksi
  - Model meramalkan bagaimana suatu sistem akan bekerja

Salah satu perangkat pemodelan adalah Unified Modeling Language (UML).

5.3 Unified Modeling Language (UML)

5.3.1 Pengenalan UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrogramana prosedural atau struktural, kemudian juga

ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan. Secara analogi jika dengan bahasa yang kita gunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar.

5.3.2 Sejarah Singkat UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya.

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman beorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Secara aktual, penggunaan bahasa pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak menarik perhatian di saat itu. Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh

Sally Shlaer dan Stephen Mellor (Shlaer-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon, 1991), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991), dan masih banyak lagi. Buku terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang fokus pada *use case* dan proses pengembangan. Sekitar lima tahun kemudian muncul buku yang membahas mengenai metodologi berorientasi objek yang diikuti dengan buku-buku yang lainnya. Di dalamnya juga membahas mengenai konsep, definisi, notasi, terminologi, dan proses mengenai metodologi berorientasi objek.

Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep Object Modelling Technique (OMT) dari Rumbaugh dan Booch (1991), konsep The Classes, Responsibilities, Collaborators (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama Rational Software Corporation menghasilkan bahasa yang disebut dengan Unified Modeling Language (UML). Pada 1996, Object Management Group (OMG) mengajukan proposal agar adanya standardisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya.

5.3.3 View dan Diagram UML

Tidak ada batasan yang jelas antara aneka ragam konsep dan kosntruksi di dalam UML, tapi untuk pemahaman yang lebih mudah, UML dibagi menjadi beberapa *view* atau pandangan. *View* atau pandangan adalah bagian yang simpel dari konstruksi pemodelan UML yang merepresentasikan aspek dari sebuah sistem. Pembagian menjadi *view* atau pandangan yang berbeda bukanlah sesuatu yang baku tergantung dari kebutuhan, tapi diharapkan dengan adanya *view* akan memudahkan kosntruksi UML. Satu atau lebih diagram merepresentasikan konsep notasi visual pada setiap *view* atau padangan.

Pada level atas, *view* atau pandangan dapat dibagi menjadi tiga area:

- klasifikasi struktural (*structural clasification*)
  mendeskripsikan hubungan segala hal yang ada di dalam sistem
- kelakuan dinamik (*dynamic behavior*)
  mendeskripsikan kelakuan sistem, atau urutan perubahan yang dialami sistem
- pengelolaan model (*model management*).
  mendeskripsikan keterkaitan organisasi dengan hirarki unit yang ada di dalam sistem

Berikut adalah keterkaitan antara *view* dan diagram di dalam UML:

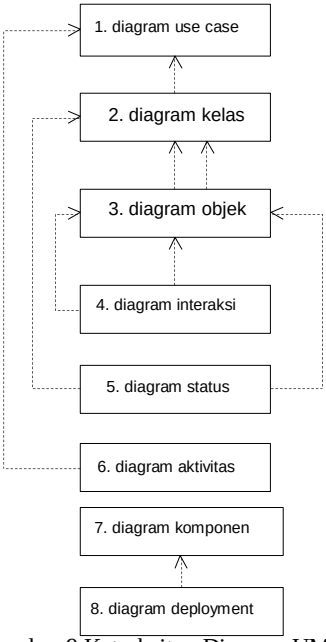
Area Mayor	View	Diagram
struktural	*static view* *view* atau pandangan yang tidak bergantung pada waktu	diagram kelas
	use case view *view* atau pandangan dari segi fungsionalitas sistem	diagram *use case*
	implementation view *view* atau pandangan dari segi komponen implementasi sistem	diagram komponen
	deployment view *view* atau pandangan dari segi *node* tempat komponen di-*deploy*	diagram *deployment*

Area Mayor	View	Diagram
dinamik	*state machine view* *view* atau pandangan dari segi status yang dialami sistem berdasarkan objek-objek sistem	diagram status
	activity view *view* atau pandangan dari segi aktivitas yang dilakukan oleh sistem	diagram aktivitas
	Diagram interaksi	diagram sekuen
		diagram kolaborasi
pengelolaan model (*model-management*)	*model-management view* *view* atau pandangan dari segi pengelolaan model sistem	diagram kelas

5.3.4 Langkah-langkah pembuatan UML

UML merupakan diagram yang saling terkait oleh karena itu perlu adanya kekonsistenan rancangan diagram yang satu dengan lainnya, bukan asal menggambar.

Berikut adalah keterkaitan diagram-diagram pada UML beserta urutan pembuatannya.



Gambar 9 Keterkaitan Diagram UML

Keterangan:

⋯⋯> terkait dengan diagram sebelumnya

Urutan dan keterkaitan antar diagram akan dibahas pada bab-bab selanjutnya beserta cara penggambaran diagram-diagram UML.

5.4 Pengertian Use case

Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan. Terdapat dua jenis kebutuhan, yaitu kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan pengguna dan *stakeholder* sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna dan *stakeholder*. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional. (Nick Jenkins, 2005).

Kebutuhan fungsional akan digambarkan melalui sebuah diagram yang dinamakan diagram use case. *Use Case Diagram* atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

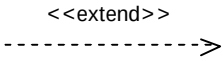
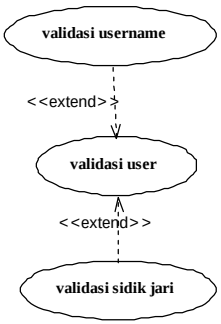
Terdapat beberapa simbol dalam menggambarkan diagram use case, yaitu *use cases*, aktor dan relasi. Simbol ini akan dijelaskan pada sub bab 6.2.

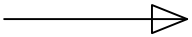
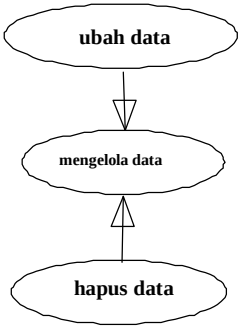
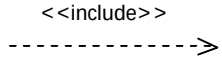
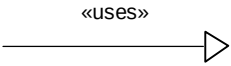
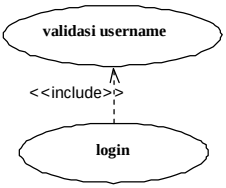
Hal yang perlu diingat mengenai diagram use case adalah diagram use case bukan menggambarkan tampilan antarmuka (*user interface*), arsitektur dari sistem, kebutuhan nonfungsional, dan tujuan performansi. Sedangkan untuk penamaan *use cases* adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja.

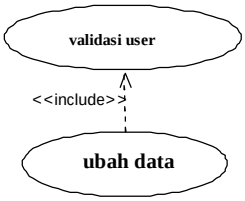
5.5 Simbol-simbol pada Use case

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Simbol	Deskripsi
Use case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor / *actor*	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang;

Simbol	Deskripsi
	biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / *association*	komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* 	relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  arah panah mengarah pada *use case* yang ditambahkan
Generalisasi / *generalization*	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case*

Simbol	Deskripsi
	dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
Menggunakan / *include* / *uses*   	relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini ada dua sudut pandang yang cukup besar mengenai include di *use case*: ■ include berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut: 

Simbol	Deskripsi
	include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. arah panah include mengarah pada *use case* yang dipakai

Arah panah relasi pada *use case* mengarah pada *use case* yang lebih besar kontrolnya atau yang dipakai.

5.6 Menemukan aktor

Pekerjaan awal dalam mendisain sistem adalah menemukan aktor, menemukan fungsionalitas dan membatasi sistem yang akan dibuat. Pembatasan sistem ini penting untuk menemukan aktor. Karena dari sinilah kita akan menentukan apakah sesuatu itu adalah aktor dan apakah aktor tersebut akan berbentuk orang atau sistem lain.

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu (Kurt Bittner, Ian Spence. 2002).

Dilihat dari pengertiannya, yang perlu anda pahami adalah memisahkan sistem yang akan dibangun dengan yang ada di luar sistem. Oleh karenanya, anda perlu membatasi sistem yang akan dibuat dan segala sesuatu yang beriteraksi dengan sistem adalah aktor.

Cara mudah untuk menemukan aktor adalah dengan bertanya hal-hal berikut:

- SIAPA yang akan menggunakan sistem?
- APAKAH sistem tersebut akan memberikan NILAI bagi aktor?

Namun, yang perlu diingat adalah tidak semua aktor adalah manusia, bisa saja sistem lain yang berinteraksi dengan sistem yang anda buat. Untuk menemukan sistem lain sebagai aktor, hal-hal di bawah ini bisa menjadi pertimbangan

- Jika anda bergantung pada sistem lain untuk melakukan sesuatu, maka sistem lain itu adalah aktor.
- Jika sistem lain itu meminta (*request*) informasi dari sistem anda, maka sistem lain itu adalah aktor

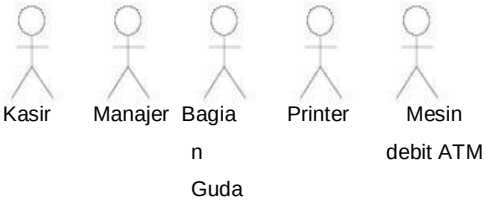
Untuk penamaan aktor diberi nama sesuai dengan PERAN-nya.

Contoh, pada sistem pencatatan penjualan di Supermarket.

Pertanyaan	Analisis
Siapa sajakah yang berinteraksi dengan sistem pencatatan penjualan di supermarket?	■ Bagian yang akan mencatat penjualan barang ■ Bagian yang ingin tahu berapa besar keuntungan yang didapatkan ■ Bagian yang ingin tahu berapa banyak produk yang berkurang
Peran apa saja yang terlibat?	Kasir, manajer, bagian gudang.
Nilai apa sajakah yang akan diberikan sistem kepada aktor?	Nilai bagi kasir: ■ Ia akan mendapatkan struk belanja. ■ Lama aktivitas kerja akan terekam kedalam sistem. Nilai bagi manajer ■ Ia perlu mengetahui laporan

	keuntungan dalam rentang waktu tertentu Nilai bagi bagian gudang ■ Ia perlu mengetahui produk apa saja yang berkurang
Apakah sistem pencatatan penjualan bergantung pada sesuatu?	Printer ■ Untuk mencetak struk Mesin debit ATM ■ Untuk menarik sejumlah uang pada *account* seseorang

Jadi, aktor yang ada pada sistem pencatatan penjualan supermarket adalah



Jika anda perhatikan dari tabel di atas, pertanyaan yang akan muncul adalah mengapa struk belanja menjadi nilai bagi kasir, dan bukannya pelanggan? Struk belanja memang nilai bagi pelanggan, namun yang perlu diingat adalah pelanggan tidak berinteraksi langsung dengan sistem, kasir-lah yang berinteraksi langsung dengan sistem. Pelanggan akan mendapatkan nilainya melalui kasir.

Sistem dibangun untuk menyediakan kebutuhan bagi aktor, jika suatu saat nanti *stakeholder* akan menentukan bahwa sistem pencatatan penjualan akan berinteraksi dengan pelanggan, maka aktor di atas pun tentu saja akan berubah. Inilah yang dimaksud dengan batasan sistem. *Stakeholder* dan pengguna akan menentukan batasan sistem yang akan dibuat.

5.7 Menemukan use case

Jika anda sudah berhasil menemukan aktor, maka untuk menemukan use case akan lebih mudah dilakukan. Sebuah use case harus mendeskripsikan sebuah pekerjaan dimana pekerjaan tersebut akan memberikan NILAI yang bermanfaat bagi aktor (Kurt Bittner, Ian Spence. 2002).

Pengertian ini penting untuk diingat, karena dari hal inilah akan menentukan bahwa sebuah use case tidak akan menjadi terlalu kecil. Karena use case yang terlalu kecil tidak akan memberikan nilai bagi aktor.

Untuk menemukan use cases, mulailah dari sudut pandang aktor, misalnya dengan bertanya

- Informasi apa sajakah yang akan didapatkan aktor dari sistem?
- Apakah ada kejadian dari sistem yang perlu diberitahukan ke aktor?

Sedangkan dari sudut pandang sistem, misalnya dengan pertanyaan sebagai berikut

- Apakah ada informasi yang perlu disimpan atau diambil dari sistem?
- Apakah ada informasi yang harus dimasukkan oleh aktor?

Setiap use case harus dijelaskan alur prosesnya melalui sebuah deskripsi use case (*use case description*) atau scenario use case. Deskripsi use case berisi:

- Nama use case yaitu penamaan use case yang menggunakan kata kerja
- Deskripsi yaitu penjelasan mengenai tujuan use case dan nilai yang akan didapatkan oleh aktor
- Kondisi sebelum (*pre-condition*) yaitu kondisi-kondisi yang perlu ada sebelum use case dilakukan.
- Kondisi sesudah (*post-condition*) yaitu kondisi-kondisi yang sudah dipenuhi ketika uses case sudah dilaksanakan
- Alur dasar (*basic flow*) yaitu alur yang menceritakan jika semua aksi yang dilakukan adalah benar atau proses yang harusnya terjadi
- Alur alternatif (*alternatif flow*) yaitu alur yang menceritakan aksi alternatif, yang berbeda dari alur dasar.

Kesalahan yang sering muncul di diagram use case (Kurt Bittner, Ian Spence. 2002)

Seringkali sebuah use case dianggap sebagai sebuah "*function*" atau item menu. Hal ini adalah salah. Perhatikan contoh berikut:

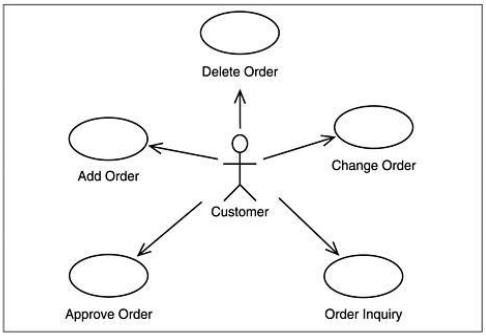


Diagram use case pemesanan.
Diambil dari Kurt Bittner, Ian Spence. 2002.

Use case di atas menggambarkan mengenai apa yang harus dilakukan oleh sistem yang terdiri dari beberapa proses yaitu menyetujui pemesanan, memesan informasi, mengubah pemesanan, menghapus pemesanan, dan menambah pemesanan. Sebenarnya, diagram di atas memperlihatkan proses penguraian fungsi-fungsi (*functional decomposition*) yaitu mengurai proses kedalam bagian yang lebih kecil. Hal ini adalah salah karena use case di atas tidak memberikan nilai kepada aktor.

Diagram use case adalah sebuah diagram yang menjelaskan apa yang harus dilakukan oleh sistem pada level konseptual sehingga kita akan memahami apakah keputusan yang diambil oleh sistem adalah benar atau tidak. Cobalah bertanya seperti ini: Apakah saya akan menggunakan proses mengubah pemesanan jika saya tidak pernah melakukan pemesanan? Tentu saja tidak. Semua proses di atas akan menjadi berguna jika terdapat proses melakukan pemesanan, dan semua proses di atas sebenarnya berkaitan dengan melakukan pemesanan.

Apa yang salah dari diagram di atas? Diagram di atas tidak memberikan nilai kepada aktor, atau dengan kata lain jika kita menggambarkan diagram seperti di atas, nilai akan menjadi hilang. Sebuah use case seharusnya dibuat untuk menghasilkan suatu nilai kepada aktor, pada level tertentu jika aktor melakukan pemesanan maka proses tersebut akan memberikan nilai kepada

aktor. Tapi jika proses pemesanan saja tidak pernah dilakukan, apakah hal ini akan memberikan nilai? Tentu saja tidak.

Oleh karena itu, gambarlah diagram use case yang berfokus pada nilai yang akan diberikan kepada aktor. Sehingga diagram use case di atas dapat diubah menjadi sebagai berikut

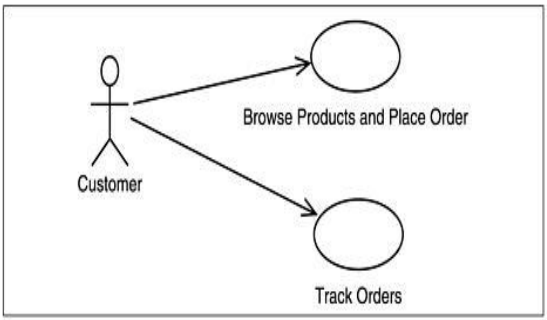


Diagram use case pemesanan.
Diambil dari Kurt Bittner, Ian Spence. 2002.

5.8 Studi Kasus

Sistem informasi manajemen perpustakaan merupakan sebuah sistem informasi untuk mengelola informasi yang diperlukan dalam suatu perpustakaan yang meliputi pendaftaran pustaka, anggota, dan proses peminjaman pustaka. Aturan perpustakaan yang harus diatasi pada sistem informasi manajemen perpustakaan yang akan dimodelkan adalah sebagai berikut:

1. Pustaka dapat memiliki lebih dari satu pengarang
2. Anggota dapat memiliki lebih dari satu nomor telepon
3. Seorang anggota dapat melakukan sebuah peminjaman dalam satu waktu dan boleh lebih dari satu pustaka
4. Seorang anggota dapat mengembalikan pustaka yang dipinjam tidak dalam waktu yang bersamaan walaupun pustaka-pustaka itu dipinjam pada waktu yang sama.
5. Pengunjung yang bukan anggota tidak diperbolehkan meminjam pustaka.

6. Proses pendaftaran pustaka, anggota, dan peminjaman dilakukan oleh petugas perpustakaan.
7. Anggota dan pengunjung dapat melakukan pencarian pustaka.

Sistem informasi yang akan dibuat adalah aplikasi berbasis web. Manajemen perpustakaan meliputi fungsi-fungsi sebagai berikut:

1. Mengelola data pustaka, meliputi:
    a. Memasukkan data pustaka
    b. Mengubah data pustaka
    c. Menghapus data pustaka
2. Mengelola data anggota, meliputi:
    a. Memasukkan data anggota
    b. Mengubah data anggota
    c. Menghapus data anggota
3. Mengelola data peminjaman, meliputi:
    a. Memasukkan data peminjaman
    b. Mengubah data peminjaman (mekanisme pengembalian pustaka)
4. Mencari pustaka

Pemecahan studi kasus tahap pertama yaitu melakukan pencarian aktor. Seperti telah dijelaskan di sub bab 6.3, mulailah bertanya dengan SIAPA, PERAN dan NILAI apa yang akan didapatkan. Sehingga didapatkan aktor sebagai berikut:

No	Aktor	Deskripsi
1.	Petugas perpustakaan	orang yang bertugas dan memiliki hak akses untuk melakukan operasi pengelolaan data pustaka, anggota, dan proses pemiinjaman pustaka
2.	Anggota/pengunjung perpustakaan	anggota adalah orang yang diperbolehkan meminjam pustaka sesuai dengan hak aksesnya, sedangkan pengunjung hanya memiliki hak akses melihat pustaka dan membaca di perpustakaan tanpa memiliki hak untuk meminjam pustaka.

Tahap selanjutnya adalah menemukan use case. Mulailah bertanya dengan INFORMASI apa yang akan diberikan oleh sistem kepada aktor. Sehingga didapatkan use case sebagai berikut.

No	*Use case*	Deskripsi
1.	Memasukkan data pustaka	merupakan proses memasukkan data pustaka ke dalam basis data
2.	Memasukkan data anggota	merupakan proses memasukkan data anggota ke dalam basis data
3.	Memasukkan data peminjaman	merupakan proses memasukkan data peminjaman ketika ada anggota yang meminjam pustaka
4.	Mencari pustaka	mencari pustaka berdasarkan judul, nama pengarang, jenis, dan kode pustaka dimana akan menampilkan data pustaka yang dicari

Tahap ketiga adalah membuat skenario per-use case. Berikut adalah skenario jalannya masing-masing *use case* yang telah didefinisikan sebelumnya:

Nama Use Case : Memasukkan data pustaka
Aktor : Pustakawan
Deskripsi : Proses ini adalah sebuah kegiatan untuk mendaftarkan data pustaka yang baru ataupun mengubah data pustaka yang sudah ada.
Pre-condition : 1. Pustakawan sudah harus memiliki IDPustakawan agar bisa memproses data buku baru.
2. Pustakawan sudah harus berada di menu pustaka
Post-condition : 1. Data informasi buku baru sudah tersimpan

Aksi Aktor	Reaksi Sistem
Alur dasar (*basic flow*)	
1. Memasukkan data pustaka seperti judul buku, penerbit, tahun terbit, pengarang, jumlah halaman,	

Aksi Aktor	Reaksi Sistem
kondisi buku di menu memasukkan pustaka	
2. Menekan tombol “Simpan”	
	3. Mengecek valid tidaknya data masukan
	4. Jika data pustaka yang dimasukkan valid, maka data pustaka akan disimpan di database dan akan menampilkan pesan “sukses disimpan”
Alur alternatif No 4 a. Jika data pustaka yang dimasukkan tidak valid, maka akan menampilkan pesan “tidak sukses disimpan”	

Alur alternatif untuk memperbaharui data pustaka	
Aksi Aktor	**Reaksi Sistem**
1. Memasukkan judul buku atau IDBuku	
2. Menekan tombol “Cari”	
	3. Menampilkan informasi buku yang terdiri dari judul buku, penerbit, tahun terbit, pengarang, jumlah halaman, kondisi buku, letak buku
4. Memperbaharui data. Beberapa pilihan data yang dapat diperbaharui diantaranya judul buku, penerbit, tahun terbit, pengarang, jumlah halaman, kondisi buku, letak buku	
5. Menekan tombol “Simpan”	
	6. Mengecek valid tidaknya data yang diperbaharui.
	7. Jika data yang dimasukkan valid, maka data pustaka yang baru akan disimpan di database dan menampilkan pesan “sukses

Aksi Aktor	Reaksi Sistem
	disimpan"
Alur alternatif No 7 a. Jika data pustaka yang dimasukkan tidak valid, maka akan menampilkan pesan "tidak sukses disimpan"	

Alur alternatif untuk menghilangkan data pustaka	
Aksi Aktor	**Reaksi Sistem**
1. Memasukkan judul buku atau IDBuku	
2. Menekan tombol "Cari"	
	3. Menampilkan informasi buku yang terdiri dari judul buku, penerbit, tahun terbit, pengarang, jumlah halaman, kondisi buku, letak buku
4. Menekan tombol "Hapus"	
	5. Menampilkan pesan "Yakin akan dihapus?"
	6. Jika jawaban pesan adalah "Ya", maka data pustaka IDBuku yang dicari akan dihapus dari database.
Alur alternatif No 6 a. Jika jawab pesan adalah "Tidak", maka akan ditampilkan menu pustaka	

Nama *Use case* : Memasukkan data anggota
Aktor : Pustakawan
Deskripsi : Proses ini adalah sebuah kegiatan untuk mendaftarkan data anggota baru ataupun mengubah data anggota yang sudah ada.
Pre-condition : 1. Pustakawan sudah harus memiliki IDPustakawan agar bisa memproses data buku baru
2. Pustakawan sudah harus berada di menu memasukkan data anggota
Post-condition : 1. Data informasi anggota baru sudah tersimpan
Skenario:

Aksi Aktor	Reaksi Sistem
Alur dasar (*basic flow*)	
1. Memasukkan data anggota seperti NIM, nama, jurusan, angkatan, semester	
2. Menekan tombol "Simpan"	
	3. Mengecek valid tidaknya data masukan
	4. Jika data anggota valid, maka data tersebut disimpan di basis data dan akan menampilkan pesan "data anggota sudah disimpan"
Alur alternatif no 4 a. Jika data anggota tidak valid, maka sistem akan menampilkan pesan "data tidak valid"	

Alur alternatif untuk memperbaharui data anggota	
Aksi Aktor	**Reaksi Sistem**
1. Memasukkan NIM anggota perpustakaan	
2. Menekan tombol "Cari"	
	3. Menampilkan informasi anggota perpustakaan seperti nama, jurusan, angkatan, semester.
4. Memperbaharui data. Beberapa pilihan data yang dapat diperbaharui diantaranya nama, jurusan, angkatan, semester	
5. Menekan tombol "Simpan"	
	6. Mengecek valid tidaknya data yang diperbaharui
	7. Jika data yang dimasukkan valid, maka data pustaka yang baru akan disimpan di database dan menampilkan pesan "sukses disimpan"
Alur alternatif No 7 a. Jika data anggota yang diperbaharui tidak valid, maka akan menampilkan	

Aksi Aktor	Reaksi Sistem
pesan "tidak sukses disimpan"	

Alur alternatif untuk menghilangkan data anggota	
Aksi Aktor	**Reaksi Sistem**
1. Memasukkan NIM anggota perpustakaan	
2. Menekan tombol "Cari"	
	3. Menampilkan informasi anggota perpustakaan seperti nama, jurusan, angkatan, semester
4. Menekan tombol "Hapus"	
	5. Menampilkan pesan "Yakin akan dihapus?"
	6. Jika jawaban pesan adalah "Ya", maka data anggota yang dicari akan dihapus dari database.
Alur alternatif No 6 a. Jika jawab pesan adalah "Tidak", maka akan ditampilkan menu anggota	

Nama *Use case* : Memasukkan data peminjaman
Aktor : Pustakawan
Deskripsi : Proses ini adalah sebuah kegiatan untuk memasukkan data buku yang dipinjam oleh anggota perpustakaan.
Pre-condition : 1. Pustakawan sudah harus memiliki IDPustakawan agar bisa memproses data buku baru
2. Pustakawan sudah harus berada di menu peminjaman buku
Post-condition : 1. Data peminjaman buku sudah tersimpan

Aksi Aktor	Reaksi Sistem
Alur dasar (*basic flow*)	
1. Memasukkan data peminjaman diantaranya yaitu NIM anggota perpustakaan, judul buku, tanggal	

Aksi Aktor	Reaksi Sistem
peminjaman	
	2. Mengecek valid tidaknya data peminjaman
	3. Jika data peminjaman valid, maka akan disimpan ke database dan menampilkan pesan "Sukses disimpan"
Alur alternatif No 3	
Jika data peminjaman tidak valid, maka sistem akan menampilkan pesan "data tidak valid"	

Alur alternatif untuk Mengubah data peminjaman	
Aksi Aktor	**Reaksi Sistem**
Alur dasar (*basic flow*)	
1. Memasukkan NIM anggota perpustakaan	
2. Menekan tombol "Cari"	
	3. Menampilkan pustaka yang dipinjam oleh anggota perpustakaan
4. Memilih data pustaka yang akan diubah berdasarkan judul buku	
	5. Menampilkan data peminjaman yang dicari
6. Mengubah data peminjaman yang meliputi judul buku, tanggal peminjaman, tanggal pengembalian, status peminjaman	
7. Menekan tombol "Simpan"	
	8. Mengecek valid tidaknya data masukan
	9. Jika data peminjaman valid, maka akan disimpan ke dalam basis data
	10. Menampilkan pesan bahwa data sukses disimpan
Alternatif No 9	

Alur alternatif untuk Mengubah data peminjaman	
Aksi Aktor	**Reaksi Sistem**
Jika data peminjaman tidak valid, maka akan menampilkan pesan "tidak bisa disimpan"	

Nama *Use case* : Mencari pustaka
Aktor : Anggota perpustakaan
Deskripsi : Proses ini adalah sebuah kegiatan untuk menemukan pustaka sesuai dengan kriteria tertentu
Pre-condition : 1. Anggota perpustakaan sudah berada di menu pencarian
Post-condition : 1. Data pustaka ditemukan

Aksi Aktor	Reaksi Sistem
Alur dasar (*basic flow*)	
1. Memasukkan kriteria pencarian berdasarkan judul pustaka atau pengarang atau kategori pustaka	
2. Menekan tombol "Cari"	
	3. Jika data pustaka ada, maka akan menampilkan data pustaka yang dicari
Alur alternatif No 3 Jika data pustaka tidak ketemu, maka akan menampilkan pesan "Pustaka yang dicari tidak ada"	

Sehingga di dapatkan diagram use case adalah sebagai berikut

Gambar 10 Diagram *Use case* Perpustakaan

Latihan

1. Apakah yang dimaksud dengan *use case*?
2. Apa kegunaan *use case*?
3. Apakah yang dimaksud dengan <<include>> dan <<extend>> pada *use case*?
4. Bagaimana membedakan relasi asosiasi dan generalisasi pada diagram use case?
5. Buatlah diagram *use case* termasuk juga *use case scenario* pada sistem informasi apotek!

6. *Basic*
    a. Apakah yang dimaksud dengan pemodelan?
    b. Apakah kegunaan pemodelan dalam pengembangan sistem informasi?
    c. Apakah yang dimaksud dengan UML?
    d. Apa kaitan UML dengan analisis dan desain sistem informasi?
    e. Sebutkan sejarah perkembangan UML!
    f. Gambarkan dan jelaskan keterkaitan setiap diagram UML!
7. *Advanced*

Sebutkan dan jelaskan semua diagram yang ada pada UML (termasuk juga diagram yang belum disebutkan di buku ini)!

6 Diagram Kelas dan Diagram Object

Overview

Pada bab ini berisi penjelasan mengenai diagram kelas dan desain objek. Diagram tersebut digunakan untuk menggambarkan kelas-kelas yang akan digunakan pada sistem informasi. Pada bab ini, studi kasus yang digunakan sama seperti bab sebelumnya yaitu sistem informasi perpustakaan. Studi kasus tersebut juga akan digunakan pada bab-bab selanjutnya untuk memberikan gambaran menyeluruh mengenai desain sistem informasi dengan pemodelan menggunakan UML.

Tujuan

1. Mahasiswa memahami tujuan penggunaan desain kelas dan desain objek.
2. Mahasiswa memahami bagaimana membuat desain kelas yang baik.
3. Mahasiswa mampu membuat desain kelas untuk sebuah sistem informasi yang sederhana.

6.1 Pengertian Diagram Kelas

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

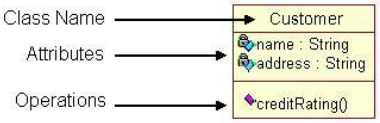
- atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
  - atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut.
- operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.

Diagram kelas menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Kelas memiliki tiga area pokok :

1. Nama
2. Atribut
3. Operasi



Contoh kelas : Manusia

- Atribut: nama, usia, tanggal lahir
- Method/Operasi: berjalan, makan, minum

6.1.1 Abstraksi Kelas

Abstraksi adalah menemukan hal-hal mendasar pada suatu objek dan mengabaikan hal-hal yang sifatnya insidental. Objek adalah instansiasi (coobtoh) dari sebuah kelas. Abstraksi bertujuan untuk menyaring *properties* dan operasi pada suatu objek, sehingga hanya tinggal yang dibutuhkan saja. Seringkali masalah yang berbeda membutuhkan sejumlah informasi yang berbeda pula pada area yang sama. Sebagai contoh, ketika kita akan membuat program untuk mengatur suatu pada objek TV dan perubhaan channel, mungkin atribut no-seri TV harus dibuang karena

tidak berguna. Tetapi ketika akan menelusuri transaksi penjualan TV, maka kita butuh nomor seri dari TV yag terjual.

6.1.2 Atribut

Atribut adalah karakteristik data yang dimiliki suatu objek dalam kelas.
Notasi dari atribut :
visibility name: type multiplicity = default {property-string}

Contoh :
- name: String [1] = "Untitled" {readOnly}
+ berarti public, - berarti private, # berarti protected
"Untitled" adalah nilai yang diberikan secara default jika tidak ditentukan saat objek dibuat
{readOnly} adalah properti tambahan dari atribut, dimana disini berarti tidak bisa dimodifikasi

6.1.3 Operasi

Operasi adalah fungsi atau transformasi yang mungkin dapat diaplikasikan ke/oleh suatu objek dalam kelas. Misalnya, suatu objek dalamkelas manusia mungkin memiliki fungsi-fungsi tersenyum, marah, makan, minum, menerima perlakuan tertentu, dan sebagainya.
Notasi dari operations
vis*ibility name (parameter-list) : return-type {property-string}*
dimana :
Parameter pada parameter-list dinotasikan seperti pada atribut

- direction name: type = default value
  Direction bisa berupa: in, out, atau inout

Contoh :
+ balanceOn (date: Date) : Money

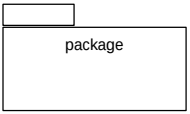
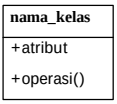
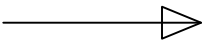
6.1.4 Multiplisitas / *Multiplicity*

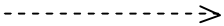
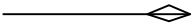
Multiplisitas menunjukkan jumlah suatu objek yang bisa berhubungan dengan objek yang lain. Umumnya ditunjukkan dengan berapa banyak objek yang bisa mengisi properti "satu" atau "banyak", tetapi secara khusus dapat ditunjukkan pula dengan bilangan integer lebih besar atau sama dengan nol.

- 1 (pasti 1)

- 0..1 (0 atau 1)
- (Tidak ada batasan, bisa 0, 1, ..., n)
- Biasanya didefinisikan batas bawah dan atas, kecuali untuk yang pasti bernilai 1

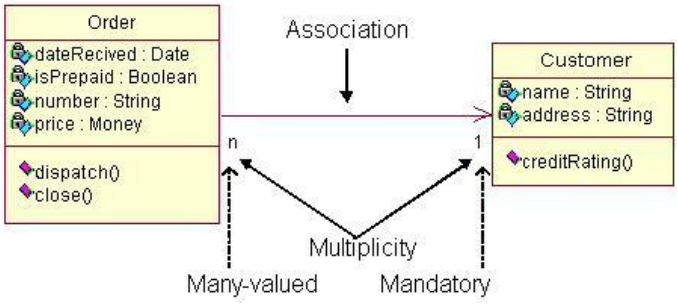
Berikut adalah simbol-simbol yang ada pada diagram kelas:

Simbol	Deskripsi
package 	*package* merupakan sebuah bungkusan dari satu atau lebih kelas
kelas - 	kelas pada struktur sistem
antarmuka / *interface* 	sama dengan konsep *interface* dalam pemrograman berorientasi objek
asosiasi / *association* 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
asosiasi berarah / *directed association* 	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
kebergantungan / *dependency*	relasi antar kelas dengan makna

Simbol	Deskripsi
	kebergantungan antar kelas
agregasi / *aggregation* 	relasi antar kelas dengan makna semua-bagian (*whole-part*)

Arah panah relasi pada diagram kelas mengarah pada diagram kelas yang lebih besar kontrolnya atau yang dipakai.

Contoh diagram kelas :



Gambar 11 Contoh Diagram Kelas

6.2 Pendefinisian Kelas pada Diagram Kelas

Kelas/*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.

Kelas menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (layanan/metoda/fungsi).

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

- Kelas main
  Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
- Kelas yang menangani tampilan sistem
  Kelas yang mendefinisikan dan mengatur tampilan ke pemakai
- Kelas yang diambil dari pendefinisian *use case*
  Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*
- Kelas yang diambil dari pendefinisian data
  Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti Koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah.

6.3 Relasi antar Kelas

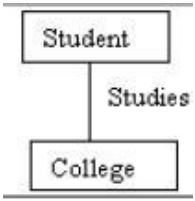
Relasi antar adalah keterkaitan hubungan antar kelas secara konseptual. UML menyediakan beberpa relasi antar kelas yang akan dijelaskan berikut ini.

6.3.1 Asosiasi

Asosiasi, yaitu hubungan statis antar kelas. Umumnya menggambarkan *class* yang memiliki atribut berupa kelas lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar kelas. Menggambarkan hubungan antar kelas

- Ditandai dengan anak panah
- Seringkali ditambahkan label dan multiplicity untuk memperjelas hubungan

Contoh asosiasi :

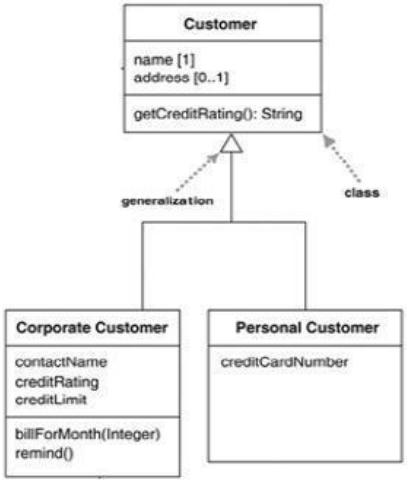


6.3.2 Agregasi

Agregasi adalah hubungan „bagian dari" atau „bagian ke keselulruhan". Suatu kelas / objek mungkin memiliki/bisa dibagi menjadi kelas/objek tertentu dimana objek/kelas yang disebut kemudian merupakan bagian dari kelas/objek yang terdahulu.

6.3.3 Generalisasi

Generalisasi adalah relasi ke atas beberapa subkelas kepada super kelas diatasnya (ditunjukkan dengan notasi segitiga). Sub kelas mewarisi fitur dari super kelasnya. Sub kelas mampu overriding metode super kelasnya.



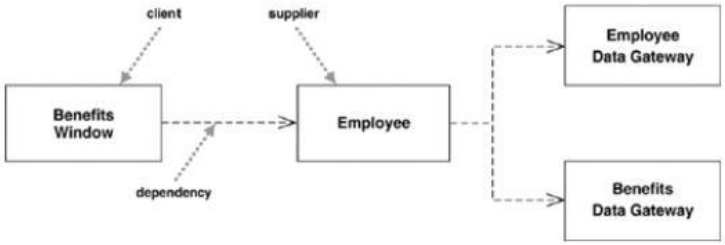
6.3.4 Dependency

Dependency adalah hubungan dimana perubahan pada suatu kelas akan mempengaruhi kelas yang lain dimana kelas yang terakhir ini bergantung pada kelas yang sebelumnya. Dalam *Dependency* antar 2 elemen jika terjadi perubahan pada salah satu elemen maka akan mengakibatkan perubahan pada elemen yang lain.

Semakin kompleks sistem, maka dependency menjadi sesuatu yang harus dipertimbangkan.

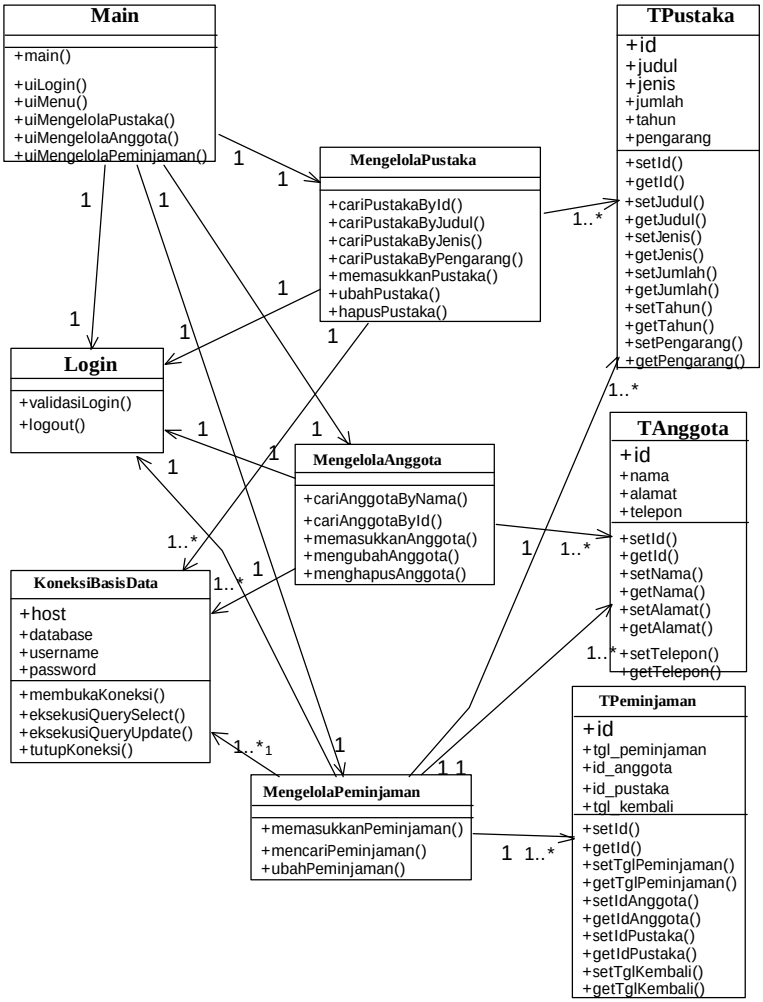
Dependency hanya berlaku satu arah. Bisa diperjelas dengan penggunaan keyword, seperti <<parameter>>, <<use>>, <<call>>

Notasi anak panah dan garis putus-putus.



6.4 Studi Kasus Diagram Kelas

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram kelas dari sistem informasi manajemen perpustakaan:



Gambar 12 Diagram Kelas Studi Kasus

Keterangan:

Nama Kelas	Keterangan
Main	merupakan kelas main yang juga merangkap sebagai kelas yang menangangi tampilan
Login	merupakan kelas proses yang diambil dari pendefinisian *use case* Login
MengelolaPustaka	merupakan kelas proses yang diambil dari pendefinisian *use case* Mengelola Pustaka yang di dalamnya harus juga menangani proses memasukkan data pustaka, mengubah data pustaka, dan menghapus data pustaka
MengelolaAnggota	merupakan kelas proses yang diambil dari pendefinisian *use case* Mengelola Anggota yang di dalamnya harus juga menangani proses memasukkan data anggota, mengubah data anggota, dan menghapus data anggota
MengelolaPeminjaman	merupakan kelas proses yang diambil dari pendefinisian *use case* Mengelola Peminjaman yang di dalamnya harus juga menangani proses memasukkan data peminjaman dan mengubah data peminjaman
TPustaka	merupakan kelas data yang digunakan untuk membungkus hasil data dari tabel TPustaka dan TPengarang
TAnggota	merupakan kelas data yang digunakan untuk membungkus hasil data dari tabel TAnggota dan TTelepon
TPeminjaman	merupakan kelas data yang

Nama Kelas	Keterangan
	digunakan untuk membungkus hasil data dari tabel TPustakaPinjam dan TPeminjaman
KoneksiBasisData	merupakan kelas utilitas untuk koneksi ke basis data dan melakukan query

6.5 Pengertian Diagram Objek

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan.

Untuk apa mendefinisikan sebuah kelas sedangkan pada jalannya sistem, objeknya tidak pernah dipakai. Hubungan *link* pada diagram objek merupakan hubungan memakai dan dipakai dimana dua buah objek akan dihubungkan oleh *link* jika ada objek yang dipakai oleh objek lainnya.

Sebuah diagram objek merupakan gambarna objek-objek dalam sebuah sistem pada satu titik waktu. Karena lebih menonjolkan perintah-perintah daripada kelas. Diagram objek sering disebut juga sebagai sebuah diagram perintah. Elemen-elemen sebuah diagram objek adalah spesifikasi perintah.

Berikut adalah simbol-simbol yang ada pada diagram objek:

Simbol	Deskripsi
Objek - nama_objek : nama_kelas atribut = nilai	objek dari kelas yang berjalan saat sistem dijalankan
Link ———	relasi antar objek

Contoh Diagram Objek

1. Dari Diagram kelas ke diagram objek

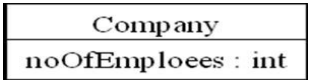
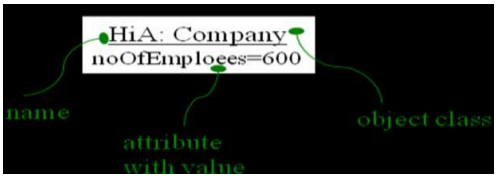
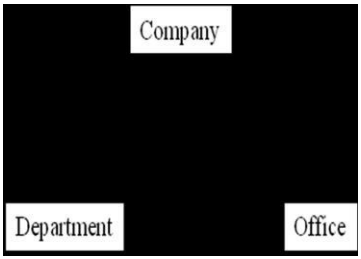


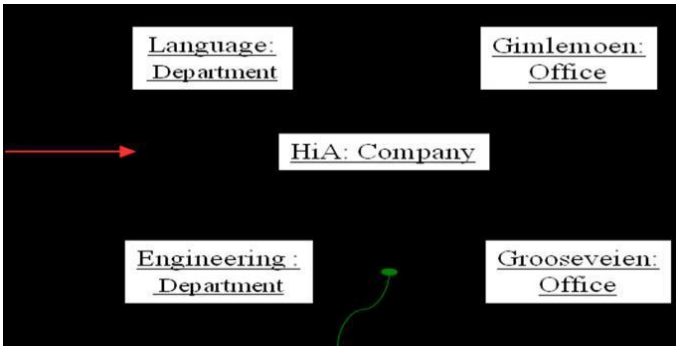
Diagram objek yang mungkin :



2, Misalkan diberikan Diagram kelas sebagai berikut :



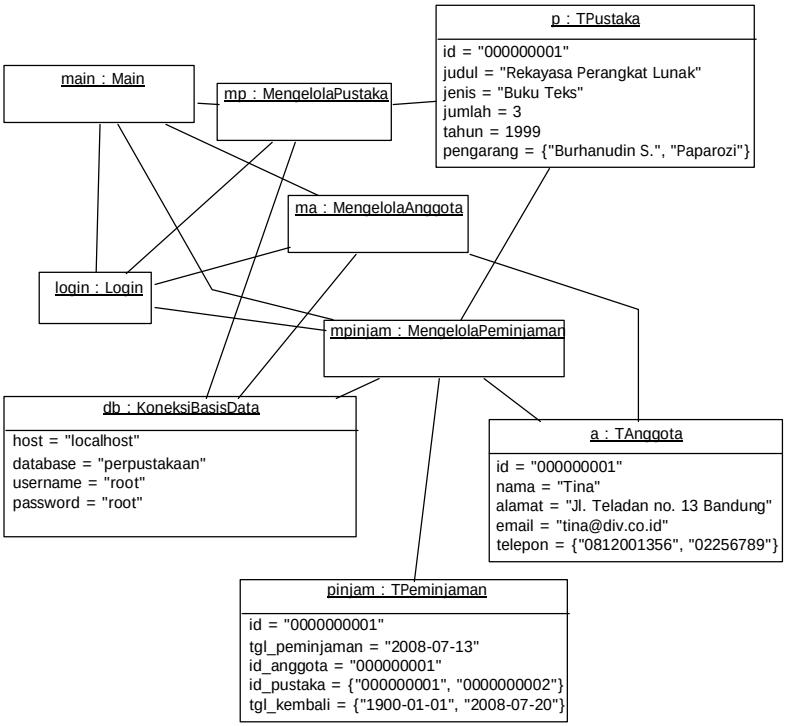
Maka kemungkinan diagram objeknya adalah :



Link - link adalah sebuah instance dari asosiasi, sama saja dengan sebuah objek menjadi sebuah instan sebuah kelas

6.6 Studi Kasus Diagram Objek

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram objek dari sistem informasi manajemen perpustakaan:



Gambar 13 Diagram Objek Studi Kasus

Latihan

1. *Basic*
   a. Apakah kegunaan diagram kelas dan diagram objek pada analisis dan desain sistem informasi?
   b. Apakah yang dimaksud dengan *dependency*, *aggregation*, dan *association* pada diagram kelas?
   c. Jenis kelas apa saja yang sebaiknya ada pada suatu sistem informasi?
2. *Advanced*

Buatlah diagram kelas untuk sistem informasi apotek!

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

7 Diagram Interaksi

Overview

Bab ini berisi mengenai cara untuk membuat diagram interaksi pada sebuah sistem informasi. Diagram interaksi digunakan untuk memodelkan interaksi antar objek dalam sebuah *use case*. Diagram interaksi merupakan diagram perilaku dari sebuah use case ketika antar objek saling berinteraksi dalam melengkapi tugas-tugasnya dan menggambarkan aliran message atau pesan. Dua jenis diagram interaksi adalah Diagram Sekuen dan Diagram Kolaborasi.

Tujuan

1. Mahasiswa memahami tujuan penggunaan diagram interaksi.
2. Mahasiswa mengetahui bagaimana cara untuk membuat diagram interaksi suatu sistem informasi.
3. Mahasiswa mampu membuat diagram interaksi untuk sebuah sistem informasi yang sederhana.

7.1 *Pengertian Diagram Interaksi*

Diagram interaksi atau *interaction diagram* digunakan untuk memodelkan interaksi objek di dalam sebuah *use case* (proses). Diagram interaksi memperlihatkan interaksi yang memuat himpunan dari objek dan relasi yang terjadi antar objek tersebut, termasuk juga bagaimana message (pesan) mengalir diantar objek. Diagram interaksi terdiri dari dua buah diagram, yaitu diagram sekuen (*sequence diagram*) dan diagram kolaborasi (*col aboration diagram*). Diagram sekuen menggambarkan urutan even dan waktu dari suatu pesan yang terjadi antar objek dalam sebuah *use case*, sedangkan diagram kolaborasi menggambarkan bagaimana objek terkoneksi secara statik (tetap) dengan penekanan pada organisasi struktural objek-objek yang mengirim dan menerima pesan.

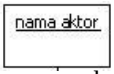
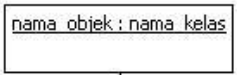
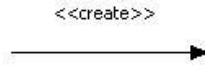
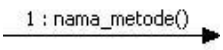
7.2 *Pengertian Diagram Sekuen*

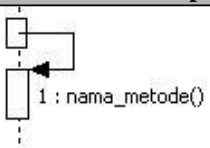
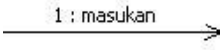
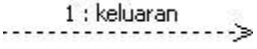
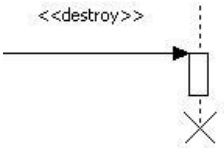
Diagram sekuen menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Simbol	Deskripsi
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan

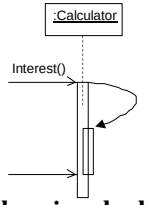
Simbol	Deskripsi
atau  tanpa waktu aktif	orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / *lifeline* 	menyatakan kehidupan suatu objek
Objek  	menyatakan objek yang berinteraksi pesan
Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe create  	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call  	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,

Simbol	Deskripsi
	 arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data/masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

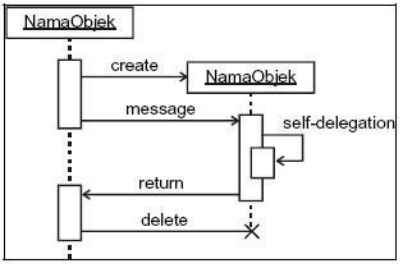
Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

Diagram Sekuen memiliki ciri yang berbeda dengan diagram interaksi pada Diagram Kolaborasi sebagai berikut :

1. Pada Diagram sekuen terdapat garis hidup objek. Garis hidup objek adalah garis tegas vertikal yang mencerminkan eksistensi sebuah objek sepanjang periode waktu. Sebagian besar objek-objek yang tercakup dalam diagram interaksi akan eksiss sepanjang durasi tertentu dari interaksi, sehingga objek-objek itu diletakkan di bagian atas diagram dengan garis hidup tergambar dari atas hingga bagian bawah diagram. Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan ***Create*** diterima suatu objek. Selain itu suatu objek juga dapat dimusnahkan dengan pesan ***Destroy***, jika kasus ini terjasi, maka garis hidupnya juga berakhir.
2. Terdapat fokus kendali (*Focus of Control*), berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi suatu objek secara langsung atau sepanjang sub ordinat. Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir dari suatu aksi (dan dapat ditandai dengan pesan ***Return***). Pada diagram ini mungkin juga memperlihatkan penyarangan (*nesting)* dan fokus kendali yang disebabkan oleh proses rekursif dengan menumpuk fokus kendali yang lain pada induknya.
   Contoh :



Gambar 9.1 Rekursi pada diagram Sekuen



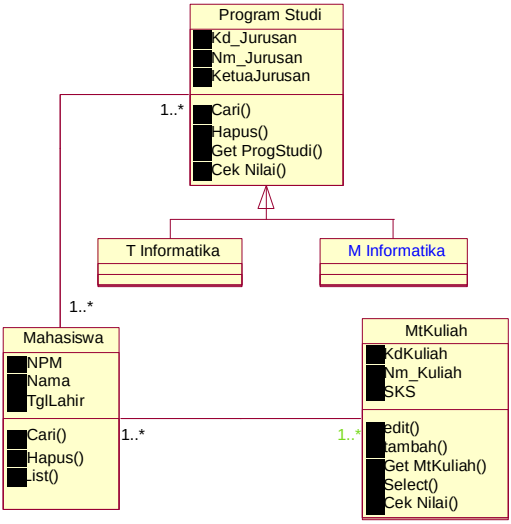
Gambar 9.2 Diagram Sekuen

7.3 *Contoh Diagram Sekuen*

Dalam kasus akademik yang memiliki program studi teknik informatika dan manajemen informatika, teridentifikasi aktor Mahasiswa dan Dosen, dengan daftar use casenya adalah :

1. Kontrak kuliah
2. Cari data
3. Tambah dt mhs
4. Edit data
5. Cek nilai

Misalkan diagram kelas hasil perancangan dari kasus akademik di atas adalah sebagai berikut :

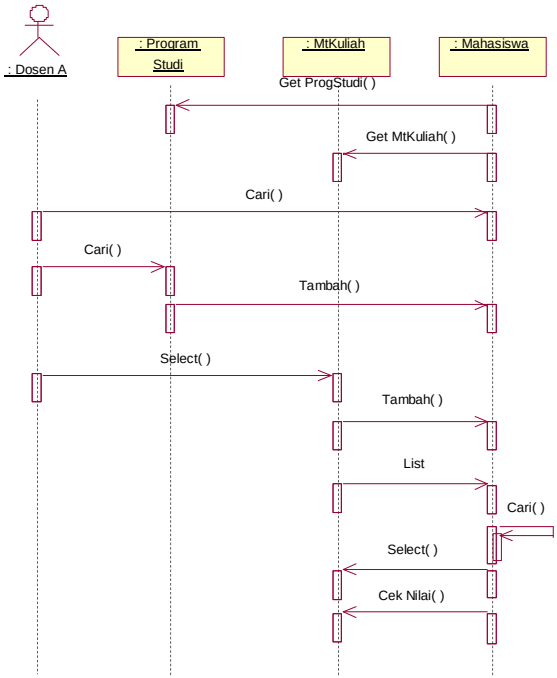


Gambar 9.3 Diagram Kelas kasus akademik

Keterangan:

Nama Kelas	Keterangan
Program Studi	merupakan kelas proses yang diambil dari pendefinisian *use case* kontrak kuliah yang memiliki spesialisasi T Informatika dan Manajemen Informatika di dalamnya

Nama Kelas	Keterangan
	harus juga menangani proses cari data, cek nilai dan hapus data
Mahasiswa	merupakan kelas proses yang diambil dari pendefinisian *use case* tambah mahasiswa yang didalamnya juga menangani proses cari, hapus, dan list.
Matakuliah	merupakan kelas proses yang diambil dari pendefinisian *use case* cari data yang di dalamnya harus juga menangani proses edit, tambah, pilih, dan cek nilai

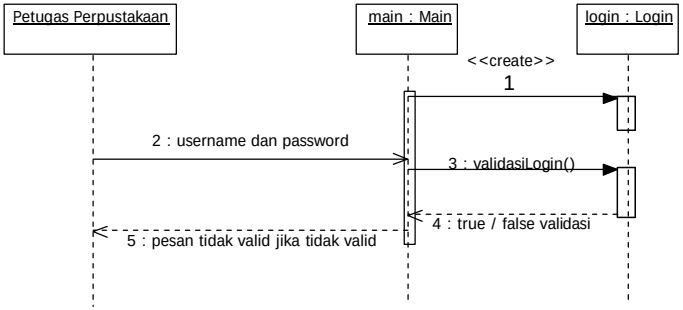


Gambar 9.4 Diagram Sekuen kasus akademik

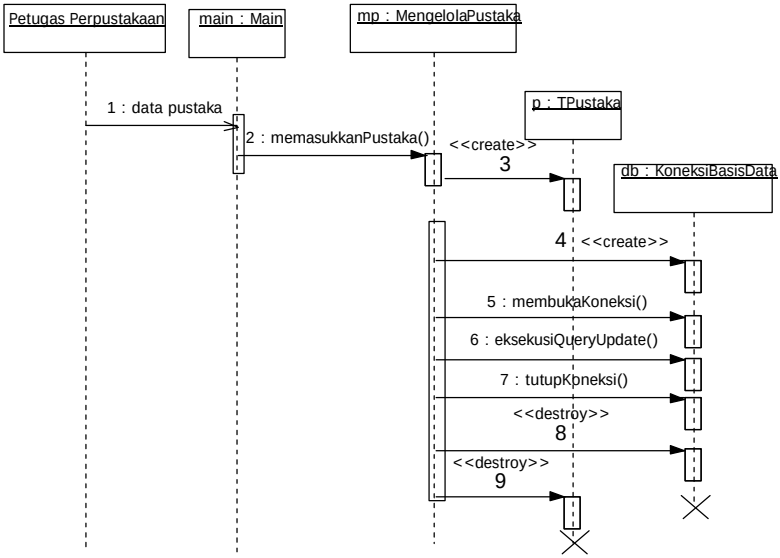
7.4 *Studi Kasus Diagram Sekuen*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram sekuen dari sistem informasi manajemen perpustakaan:

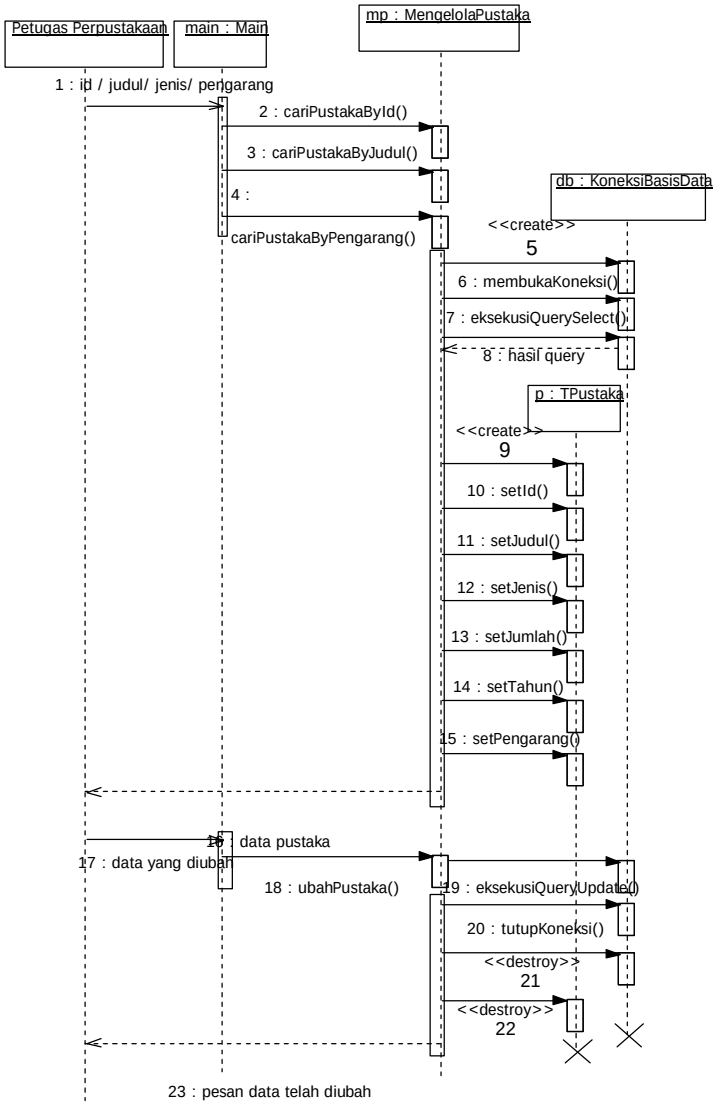
Use case: Login



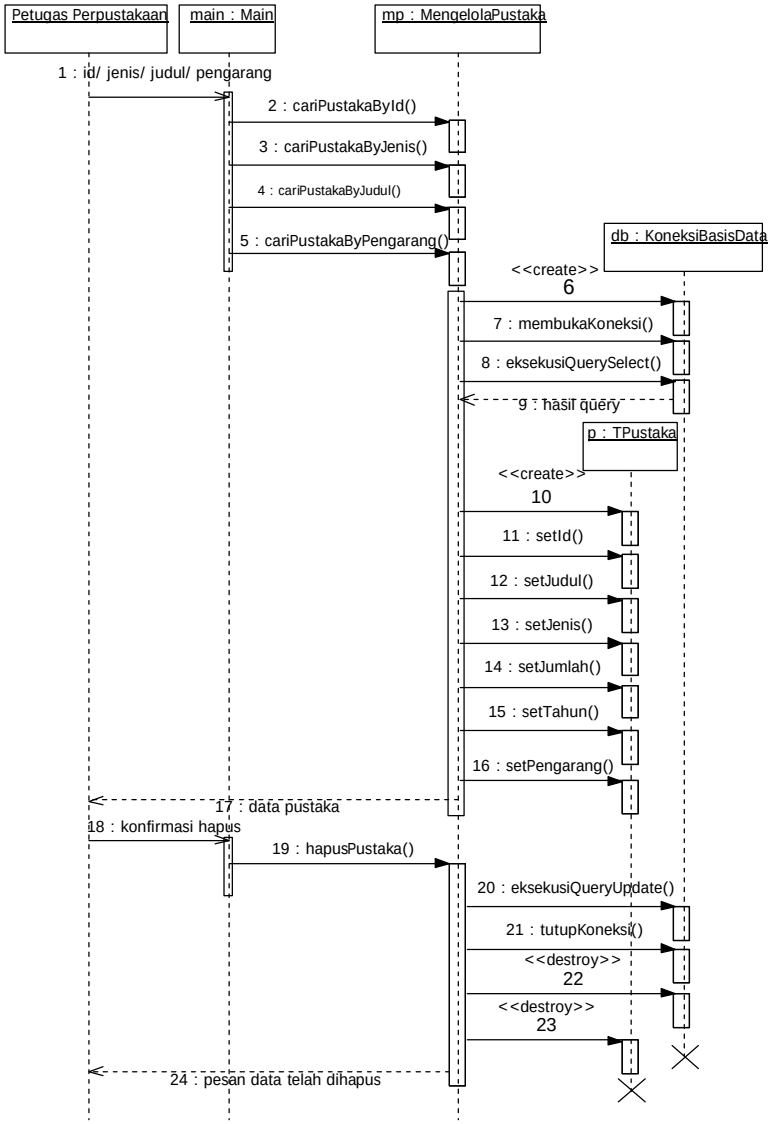
Use case: Memasukkan data pustaka



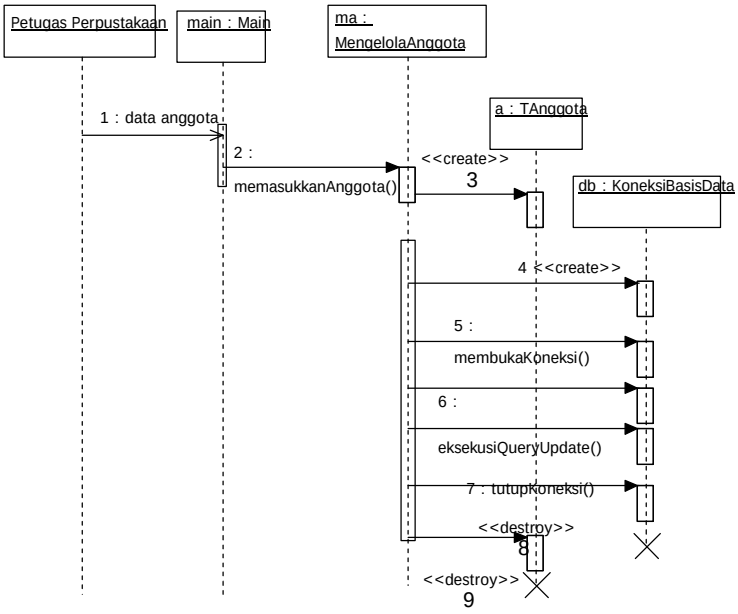
Use case: Mengubah data pustaka



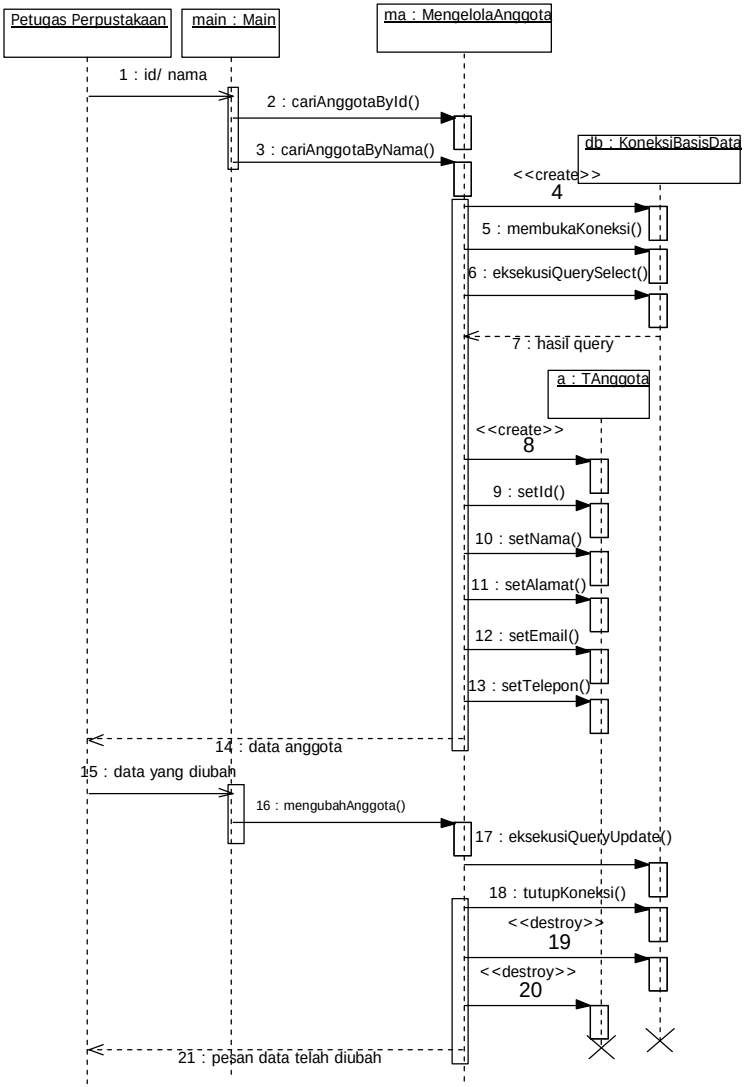
Use case: Menghapus data pustaka



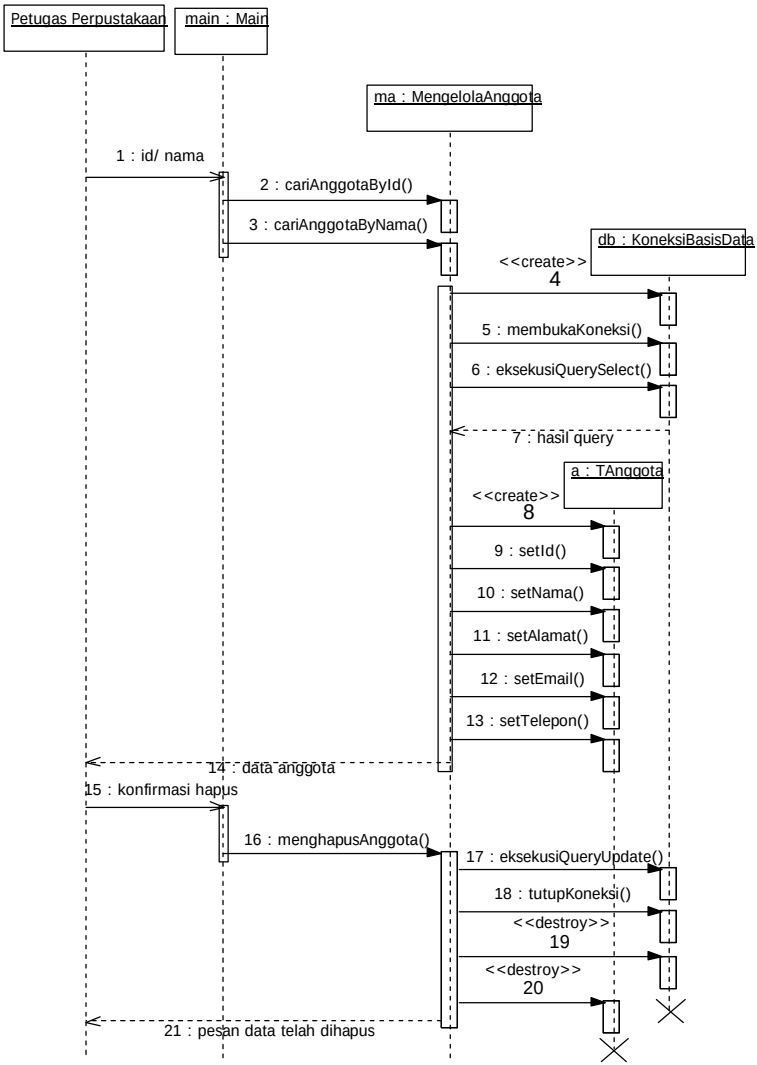
Use case: Memasukkan data anggota



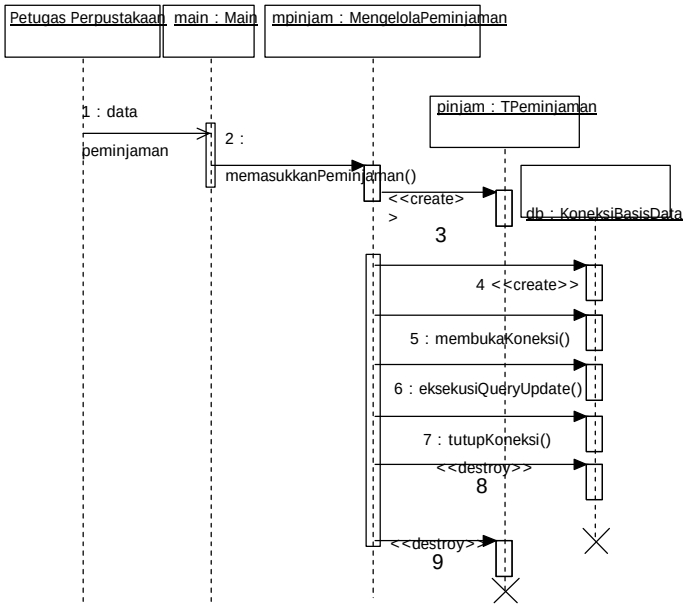
Use case: Mengubah data anggota



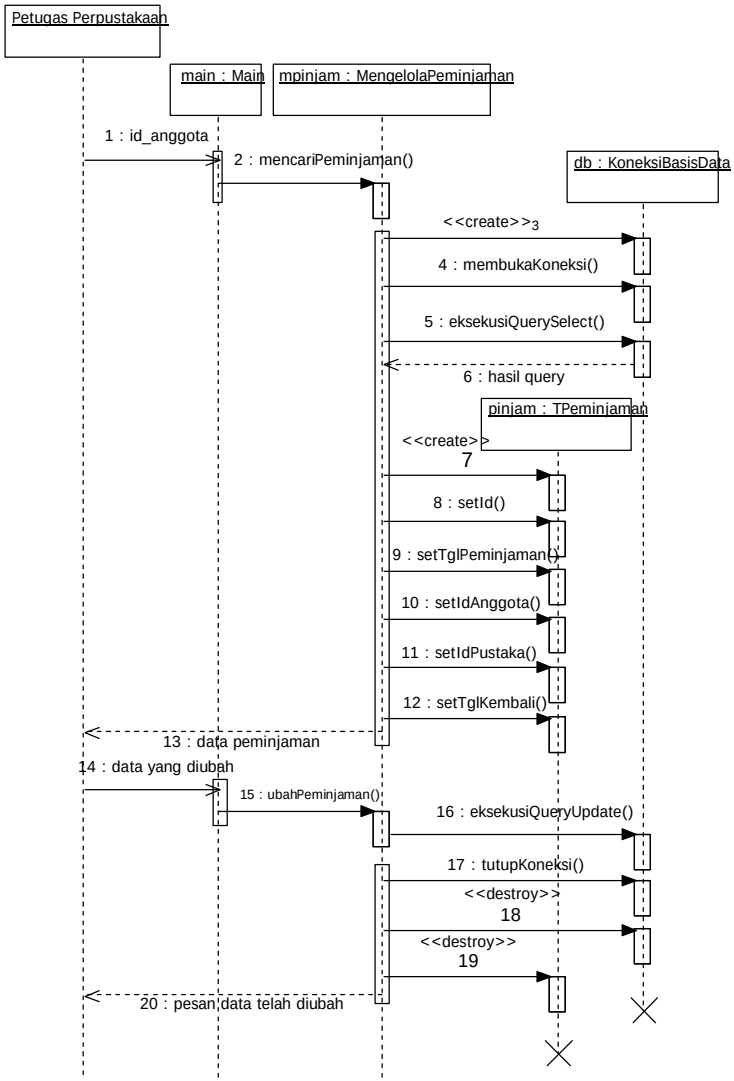
Use case: Menghapus data anggota



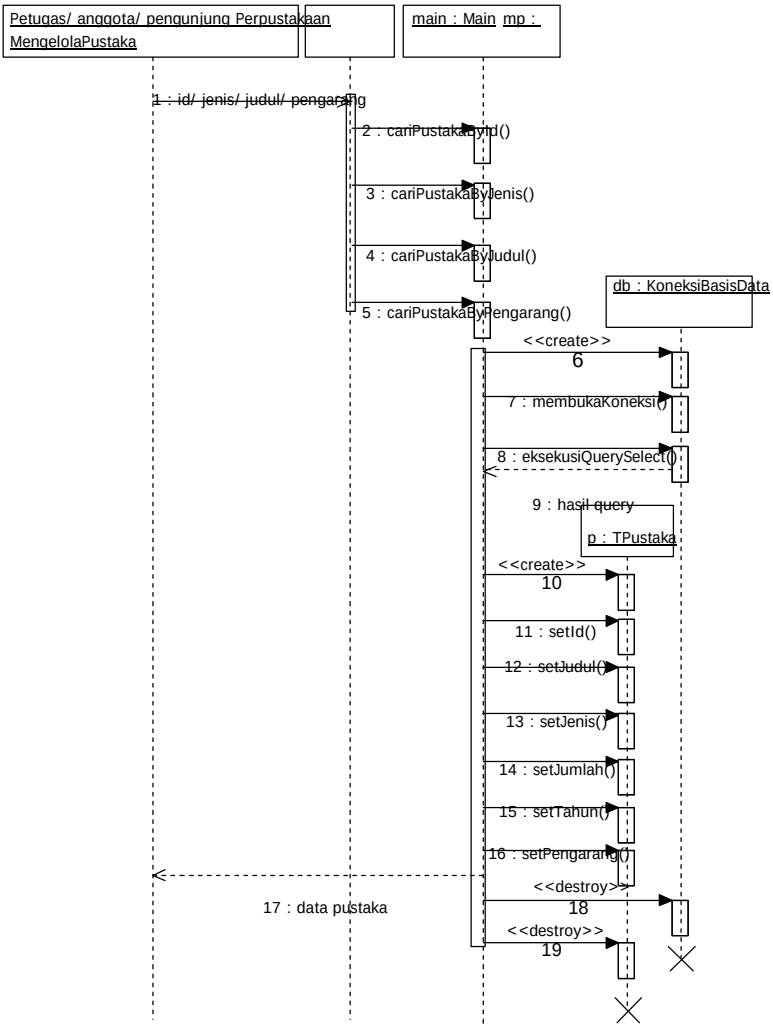
Use case: Memasukkan data peminjaman



Use case: Mengubah data peminjaman



Use case: Mencari pustaka



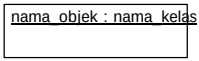
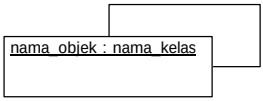
7.5 *Pengertian Diagram Kolaborasi*

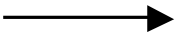
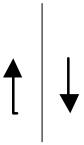
Diagram kolaborasi mengelompokkan *message* pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram kolaborasi yang dituliskan adalah operasi/metode yang dijalankan antara objek yang satu dan objek lainnya secara keseluruhan, oleh karena itu dapat diambil dari jalannya interaksi pada semua diagram sekuen. Penomororan metode dapat dilakukan berdasarkan urutan dijalankannnya metode/operasi diantara objek yang satu dengan objek lainnya atau objek itu sendiri.

Untuk menunjukkan sebuah pesan/*message*, buatlah tanda panah di dekat garis asosiasi diantara 2 objek. Arah panah menunjukkan objek yang menerima pesan. Label di dekat panah menunjukkan nomor urut dan pesanannya. Tipikal *message* meminta kepada objek yang menerimanya untuk menjalankan salah satu operasinya. Sepasang tanda kurung digunakan untuk mengakhiri message. Jika ada parameter, dapat diletakkan di dalam tanda kurung.

Dalam pemodelan kolaborasi mungkin juga ditemui sebuah objek yang mengirim pesan setelah beberapa pesan lain terkirim. Untuk ini objek harus melakukan sinkronisasi pesan dengan serangkaian pesan-pesan lainnya.

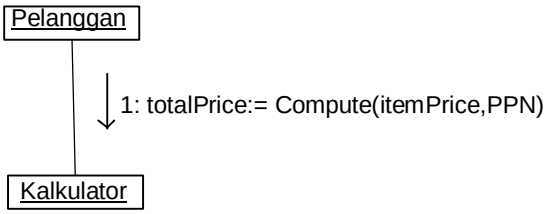
Berikut adalah simbol-simbol yang ada pada diagram kolaborasi:

Simbol	Deskripsi
Objek 	objek yang melakukan interaksi pesan
Link 	relasi antar objek yang menghubungkan objek satu dengan lainnya atau dengan dirinya sendiri 

Arah pesan / *stimulus* 	arah pesan yang terjadi, jika pada suatu *link* ada dua arah pesan yang berbeda maka arah juga digambarkan dua arah pada dua sisi *link* 

7.6 *Menunjukkan hasil pemrosesan*

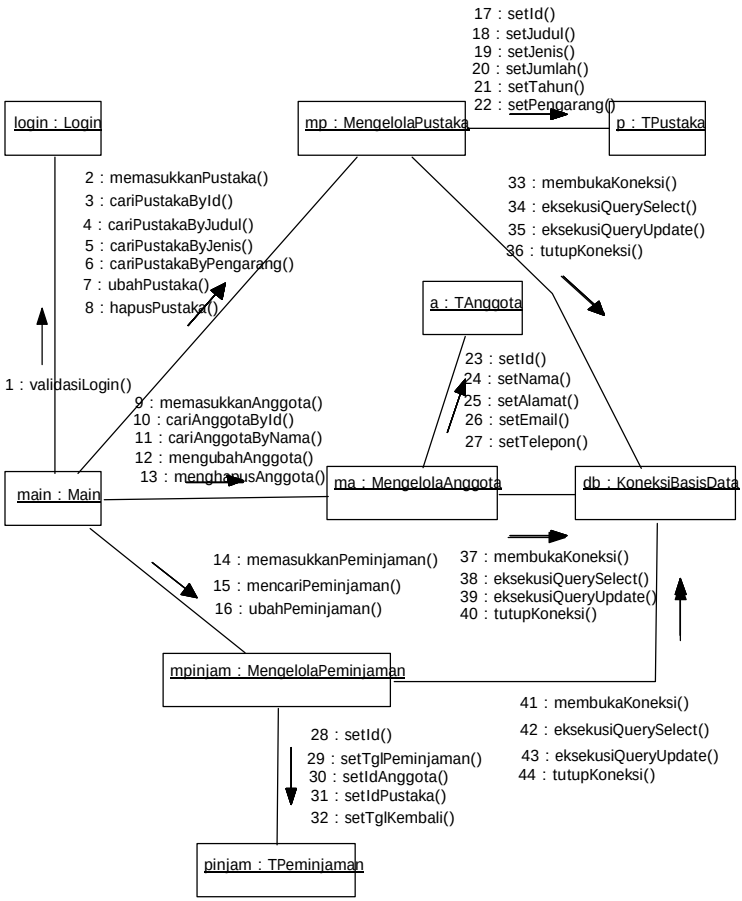
Sebuah *message* mungkin juga sebuah permintaan kepada objek untuk melakukan perhitungan dan menghasilkan sebuah nilai. Sebagai contoh adalah objek pelanggan akan meminta kepada objek kalkulator untuk menghitung harga total yang merupakan penjumlahan harga barang dan PPN. UML menyediakan sintaks untuk menuliskan hal tersebut dengan cara menuliskan nama variabel dengan tanda „ := „ diikuti dengan nama operasi dan jumlah yang dioperasikan untuk mendapatkan hasil.



Gambar 9.5 Hasil Operasi pada Kolaborasi

7.7 *Studi Kasus Diagram Kolaborasi*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram kolaborasi dari sistem informasi manajemen perpustakaan:



Gambar 14 Diagram Kolaborasi Studi Kasus

Kuis Benar Salah

1. Diagram sekuen menambahkan dimensi waktu pada interaksi di antara objek.
2. Tidak mungkin sebuah objek mempunyai sebuah operasi kepada dirinya sendiri.
3. *Message* di diagram sekuen diwakili oleh garis dengan tanda panah.
4. Simbol aktor tidak ada pada diagram sekuen.
5. Diagram kolaborasi menunjukkan pesan objek yang dikirimkan satu sama lain.
6. *Message* yang dikrimkan antar objek pada diagram kolaborasi diberi nomor urut.
7. Diagram kolaborasi bukan merupakan asosiasi di antara objek-objek.
8. Dengan diagram kolaborasi, dapat dilakukan pengiriman sebuah *message* ke banyak objek pada kelas yang sama.
9. Hubungan antar objek tidak dapat ke dirinya sendiri.
10. Diagram interaksi sama dengan diagram sekuen.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Diagram yang mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek : _____
   A. Interaksi
   B. Sekuen
   C. Kolaborasi
   D. Semua benar
   E. Tidak ada jawab
2. Yang membedakan antara Diagram sekuen dan diagram kolaborasi adalah : _____
   A. Garis hidup
   B. pesan
   C. Objek
   D. Kelas
   E. aktor
3. Pada diagram kolaborasi, jika terdapat operasi perhitungan yang menghasikan sebuah nilai, dapat digunakan tanda : _____
   A. =
   B. :
   C. :=
   D. kurung
   E. angka
4. Untuk menyatakan suatu objek mengakhiri hidup objek yang lain adalah dengan : _____
   A. Messsage
   B. Destroy
   C. Create
   D. End
   E. Semua benar
5. Diagram sekuen umumnya digunakan untuk menggambarkan suatu skenario atau urutan langkah-langkah yang dilakukan :
   A. Aktor
   B. Objek
   C. Sistem
   D. A, B, dan C benar
   E. A dan C benar

6. Diagram yang mengelompokkan pesan-pesan dalam suatu diagram adalah : _____
   A. Interaksi
   B. Sekuen
   C. Kolaborasi
   D. Semua benar
   E. Tidak ada jawab
7. Simbol pada diagram kolaborasi diantaranya : _____
   A. Objek
   B. Garis hidup
   C. Link
   D. A dan B benar
   E. A dan C benar
8. Simbol pada diagram sekuen diantaranya : _____
   A. Objek
   B. Garis hidup
   C. Link
   D. A dan B benar
   E. A dan C benar
9. Diagram sekuen harus dibuat setelah ada : _____
   A. Diagram kelas
   B. Diagram kolaborasi
   C. Model bisnis
   D. Skenario
   E. Tidak ada jawab
10. Garis tegak pada diagram sekuen menunjukkan : _____
    A. Garis hidup
    B. Waktu aktif
    C. pesan
    D. objek
    E. Semua benar

Latihan

1. *Basic*
    a. Apakah yang dimaksud dengan diagram interaksi?
    b. Apakah kegunaan diagram interaksi dalam analisis dan desain sistem informasi?
    c. Apakah yang dimaksud dengan *sequence diagram* dan *col aboration diagram*, apa perbedaan dan persamaan diantara keduanya?
    d. Apakah resiko yang dihadapi jika analisis dan desain sistem informasi tidak melakukan pemodelan dengan diagram interaksi?
    e. Apakah keterkaitan antara diagram interaksi dengan *use case* dan diagram kelas?
2. *Advanced*
    a. Buatlah *sequence diagram* untuk sistem informasi apotek!
    b. Buatlah *collaboration diagram* untuk sistem informasi apotek!

8 Diagram Status

Overview

Bab ini berisi bagaimana membuat diagram status untuk suatu sistem informasi. Diagram status digunakan untuk menyatakan kondisi (status) sebuah objek pada saat sistem informasi berjalan. Diagram interaksi dan diagram status saling melengkapi tentang perilaku dinamis sebuah sistem. Diagram interaksi menunjukkan pesan-pesan yang dilewatkan di antara objek-objek di dalam sistem selama periode waktu yang singkat. Sedangkan diagram status diagram yang menelusuri objek melalui keseluruhan siklus hidupnya.

Tujuan

1. Mahasiswa memahami tujuan penggunaan diagram status.
2. Mahasiswa mengetahui bagaimana cara untuk membuat diagram status.
3. Mahasiswa mampu membuat diagram status untuk sebuah sistem informasi yang sederhana.

8.1 *Pengertian Diagram Status*

Diagram status atau *state diagram* atau *statechart diagram* menunjukkan kondisi yang dapat dialami atau terjadi pada sebuah objek sehingga setiap objek memiliki sebuah diagram status. Diagram status diadopsi dari penggambaran kondisi mesin status (*state machine*) yang menggambarkan status apa saja yang dialami oleh mesin, misalnya mesin pembelian kopi dengan uang koin.

Diagram Status mengambarkan seluruh *state*/status yang memungkinkan obyek-obyek dalam *class* dapat dimiliki dan kejadian-kejadian yang menyebabkan satus berubah. Perubahan dalam suatu *state* disebut juga transisi (*transition*). Suatu transisi juga dapat memiliki sebuah aksi yang dihubungkan pada status, lebih spesifik apa yang harus dilakukan dalam hubungannya dengan transisi status. Pada diagram ini, perilaku sistem ditunjukkan. Sebuah status adalah kondisi selama hidup objek atau interaksi selama memenuhi suatu kondisi, melaksanakan suatu aksi, atau menunggu suatu kejadaian.

Simbol-simbol yang ada pada diagram status adalah sebagai berikut:

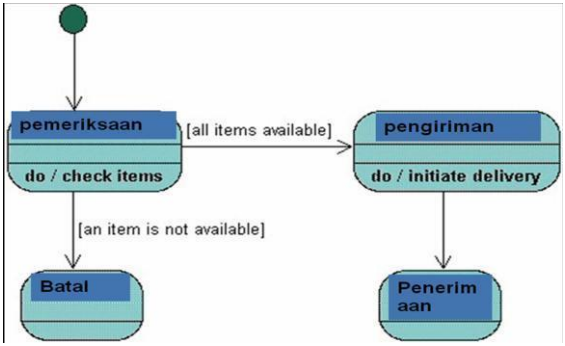
Simbol	Deskripsi
status awal / kondisi awal 	status awal alur sebuah objek, sebuah diagram status memiliki sebuah status awal
status 	status yang dialami objek selama hidupnya
status akhir / kondisi akhir 	kondisi akhir alur hidup objek, sebuah diagram status memiliki sebuah status akhir
transisi	garis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

Simbol	Deskripsi
nama transisi 	sekuen sehingga pesan pada diagram sekuen menjadi transisi bukan sebagai status, status merupakan kondisi yang dialami objek, bukan merupakan pesan (*message*), transisi juga bisa memutar pada sebuah status
	Transisi internal melingkar / ke status sendiri

8.2 *Status, Event, dan Transisi*

Objek pada sistem mengubah statusnya untuk merespon *event*/kejadian dan waktu. Secara umum, pendeteksian sebuah kejadian dapat menyebabkan sebuah objek bergerak dari satu status ke status yang lain. Keadaaan ini disebut transisi.

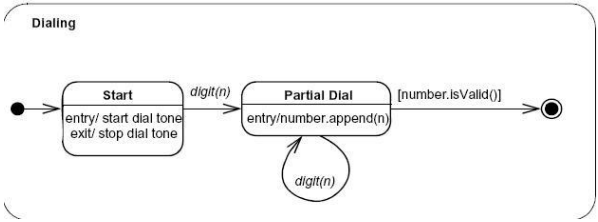
Di bawah ini contoh diagram status untuk objek Order. Sistem diawali pada status pemeriksaan yang akan melakukan kegiatan "periksa item barang." Setelah itu memeriksa apakah item tersedia atau tidak tersedia. Jika item tersedia, maka ke status pengiriman kemudian ke status penerimaan. Jika tidak tersedia maka ke status Batal.



Gambar 10.1 Diagram status sistem

8.3 Composite State

Jika diagram status akan digunakan untuk sistem yang kompleks, maka perlu penyederhanaan. Salah satu penggunaannya adalah sub status. Sub status dikelompokkan bersama-sama dalam status berdekatan karena penggunaan *properties* tertentu secara bersama-sama menjadi sebuah „super state". *Composite state* didekomposisi menjadi dua atau lebih sub status bersamaan atau menjadi sub status yang terpisah.

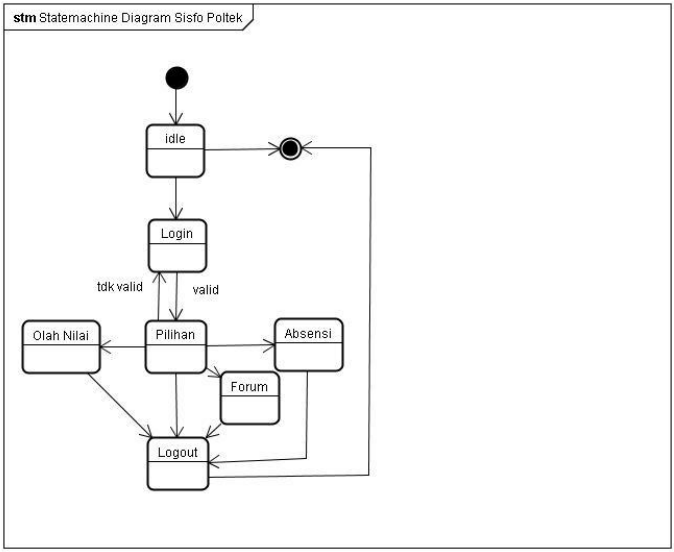


Gambar 10.2 Contoh Composite state

8.4 *Contoh Diagram Status*

Misalkan akan dibuat diagram status untuk Sistem informasi poltek online melalui http:/sisfo.poltekniktelkom.ac.id yang dioperasikan oleh dosen.
Deskripsi :
Ketika dosen melakukan login ke sistem akan divalidasi user name dan password, jika valid akan muncul tampilan pilihan olah nilai, absensi, forum diskusi atau logout.

Gambar di bawah ini menunjukkan perilaku hal di atas.

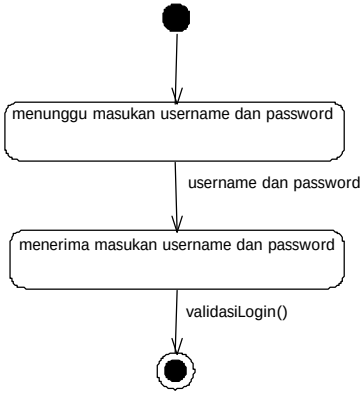


Gambar 10.2 Diagram status Sisfo Poltek Telkom

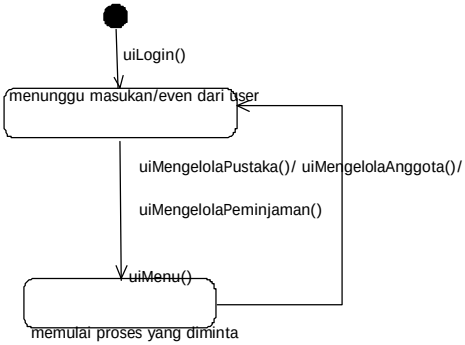
8.5 *Studi Kasus Diagram Status*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram status dari setiap objek pada diagram objek sistem informasi manajemen perpustakaan:

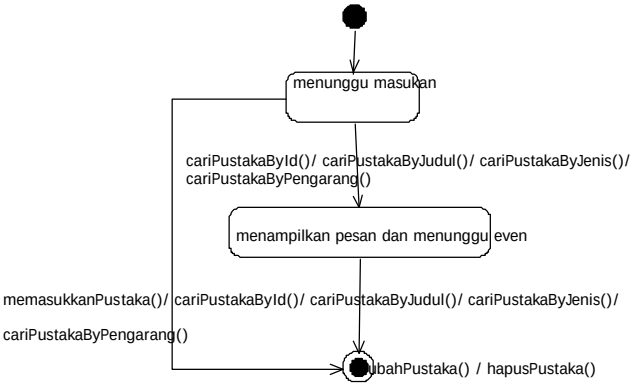
Objek: login dari kelas Login



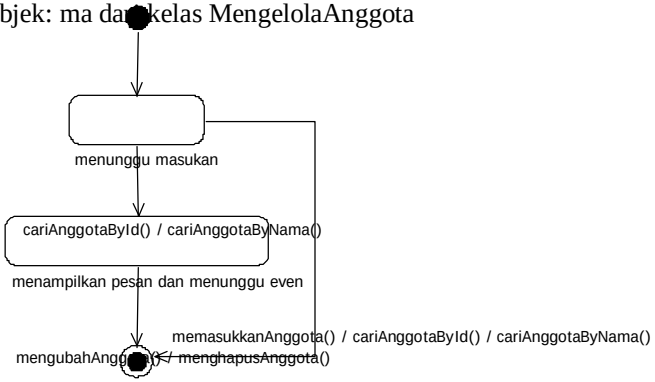
Objek: main dari kelas Main



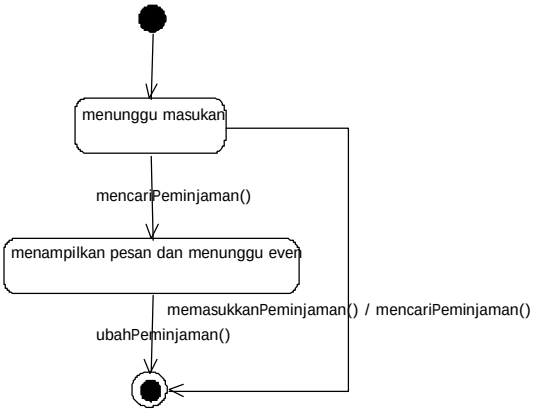
Objek: mp dari kelas MengelolaPustaka



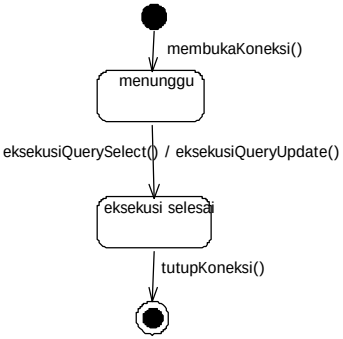
Objek: ma dari kelas MengelolaAnggota



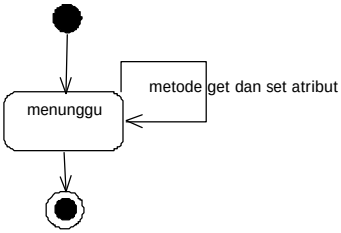
Objek: mpinjam dari kelas MengelolaPeminjaman



Objek: db dari kelas KoneksiBasisData



Objek: p dari kelas TPustaka, a dari kelas TAnggota, pinjam dari kelas TPeminjaman



Kuis Benar Salah

1. Diagram status juga menggambarkan perilaku dari sistem.
2. Perubahan status dari satu status ke status lainnya disebut *event*.
3. *Message* di diagram status diwakili oleh segiempat oval.
4. Pada diagram status, *message* tidak ditunjukkan.
5. Tidak ada simbol transisi ke diri sendiri pada diagram status.
6. Diagram status tidak menunjukkan kejadian dan kegiatan .
7. Tanda status akhir di diagram status mungkin tidak ada.
8. Mungkin saja dalam diagram status hanya terdapat satu status untuk suatu objek.
9. Di dalam suatu status mungkin terdapat satu atau lebih status lain.
10. Pesan pada diagram sekuen menjadi status bukan sebagai transisi.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Simbol lingkaran solid diarsir adalah : _____

   A. awal
   B. akhir
   C. status
   D. event
   E. transisi

2. Simbol segiempat oval adalah : _____

   A. awal
   B. akhir
   C. status
   D. event
   E. transisi

3. Simbol lingkaran seperti mata adalah : _____

   A. awal
   B. akhir
   C. status
   D. event
   E. transisi

4. Objek yang bergerak dari satu status ke status yang lain adalah : _____

   A. awal
   B. akhir
   C. status
   D. event
   E. transisi

5. Berikut adalah komponen dari diagram status, kecuali :

   A. Titik awal
   B. Titik akhir
   C. status
   D. transisi
   E. aktor

6. Diagram status memungkinkan dekomposisi status ke status yang lebih sederhana, hal ini disebut : _____

   A. State
   D. transisi

B. composite
C. Sub state
D. state
E. Tidak ada jawab

7. Transisi untuk memodelkan event sebuah objek pada status yang sama disebut : _____
   A. internal
   B. Bukan transisi
   C. action
   D. state
   E. Salah semua
8. Diagram status sesuai untuk menggambarkan perilaku : _____
   A. Objek
   B. Kolaborasi objek
   C. Use case
   D. kelas
   E. Salah semua
9. Diagram status sebaiknya digabungkan dengan diagram lain seperti : _____
   A. Diagram sekuen
   B. Diagram kolaborasi
   C. Diagram use case
   D. A dan B benar
   E. Tidak ada jawab
10. Garis melingkar pada diagram status menunjukkan : _____
    A. transisi
    B. status
    C. pesan
    D. objek
    E. Semua salah

Latihan

1. *Basic*
    a. Apakah yang dimaksud dengan *state diagram*?
    b. Apa kegunaan *state diagram*?
2. *Advanced*
    a. Kapan (dalam kondisi seperti apa) *state diagram* sebaiknya digunakan?
    a. Buatlah *state diagram* untuk sistem informasi apotek!

9 Diagram Aktivitas

Overview

Bab ini berisi cara untuk membuat diagram aktivitas. Diagram aktivitas digunakan untuk menggambarkan proses bisnis (alur kerja) suatu sistem informasi.

Sebuah Diagram aktivitas menunjukkan suatu alur kegiatan secara berurutan. Diagram aktivitas digunakan untuk mendiskripsikan kegiatan-kegiatan dalam sebuah operasi meskipun juga dapat digunakan untuk mendeskripsikan alur kegiatan yang lainnya seperti *use case* atau suatu interaksi.

Tujuan

1. Mahasiswa memahami tujuan penggunaan diagram aktivitas.
2. Mahasiswa mengetahui bagaimana cara untuk membuat diagram aktivitas sebuah sistem informasi.
3. Mahasiswa mampu membuat diagram aktivitas sebuah sistem informasi yang sederhana.

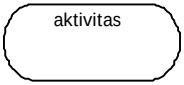
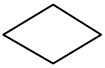
9.1 *Pengertian Diagram Aktivitas*

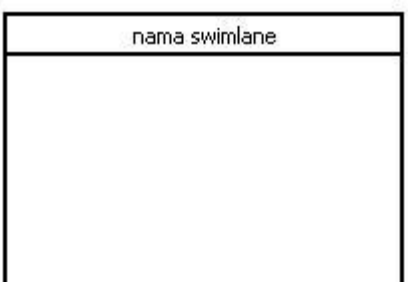
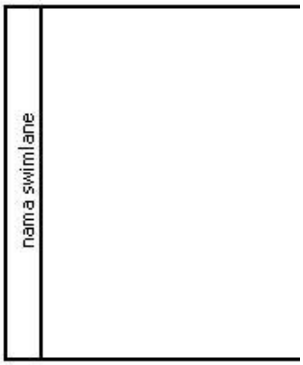
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas mendukung perilaku paralel.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

- rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
- urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
- rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Simbol	Deskripsi
status awal 	status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
aktivitas 	aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
percabangan / *decision* 	asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
penggabungan / *join*	asosiasi penggabungan dimana lebih dari satu aktivitas

Simbol	Deskripsi
	digabungkan menjadi satu
status akhir 	status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
swimlane  atau 	memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
	*fork*, digunakan utk menunjukkan kegiatan yg dilakukan secara

Simbol	Deskripsi
	paralel
	*join*, digunakan utk menunjukkan kegiatan yg digabungkan

9.2 *Membuat Diagram Aktivitas*

9.2.1 Pengantar

Diagram aktivitas mendeskripsikan aliran kerja dari perilaku sistem. Diagram ini hampir sama dengan diagram status karena kegiatan-kegiatannya merupakan status suatu pekerjaan dengan menunjukkan kegiatan yang dilakukan secara berurutan.
Sebaiknya diagram aktivitas digunakan untuk melengkapi diagram lain seperti diagram interaksi dan diagram status, karena diagram aktivitas dapat mengetahui aliran sistem yang akan dirancang. Selain itu diagram aktivitas bermanfaat untuk menganalisis use case melalui penggambaran aksi-aksi yang dibutuhkan, penggambaran algoritma berurutan yang kompleks, dan pemodelan aplikasi dengan proses paralel. Tetapi diagram aktivitas tidak menunjukkan bagaimana objek berperilaku atau objek berkolaborari secara detil.

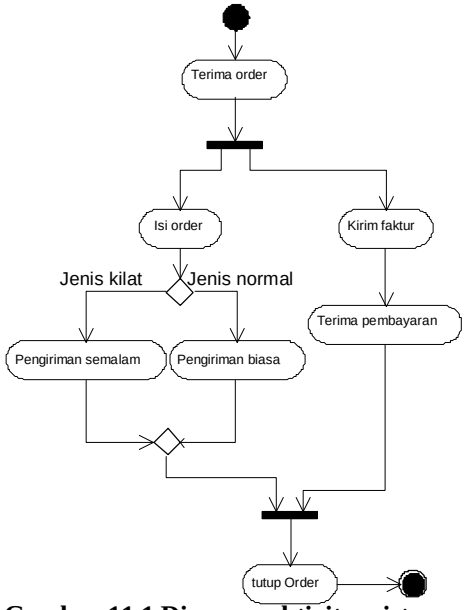
9.2.2 Langkah-langkah Penggambaran

Diagram aktivitas dibaca dari atas ke bawah, mungkin bercabang untuk menunjukkan kondisi, keputusan dan atau memiliki kegiatan paralel.
Berikut adalah langkah-langkah membuat diagram aktivitas :

1. Buat simbol status awal ketika mengawali diagram
2. Gambarkan aksi pertama dan seterusnya sesuai aliran kegiatan sistem. Gunakan sebuah *fork* ketika berbagai aktivitas terjadi secara bersamaan. Setelah penggabungan seluruh kegiatan paralel, harus digabungkan dengan simbol *join*.
3. Cabang keputusan digunakanuntuk menunjukkan suatu kegiatan yang memenuhi kondisi tertentu. Seluruh pancabangan diakhiri tanda penggabungan (mengganakan tanda decision) sebagai akhir perilaku tersebut.
4. Akhiri diagram dengan simbol status akhir

9.2.3 Contoh Diagram Aktivitas

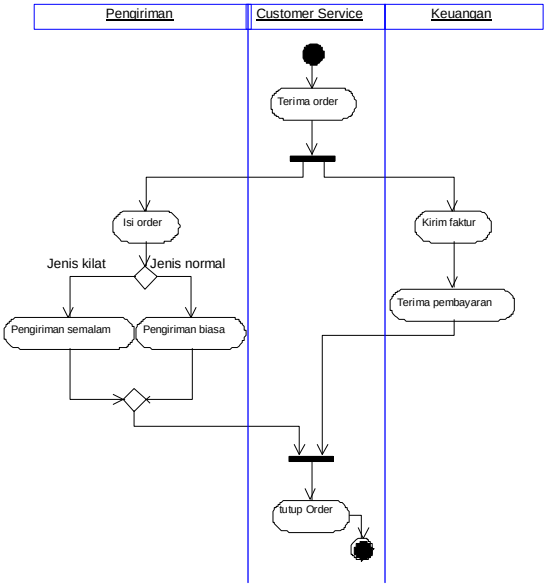
Gambar di bawah ini menunjukkan sebuah contoh sederhana dari diagram aktivitas untuk sistem Order. Diagram diawali dengan *node* status awal dan kemudian melakukan aksi terima order. Kemudian kegiatan isi order dan kirim faktur dapat dilakukan secara paralel. Setelah kirim faktur dilakukan terima pembayaran dan setelah isi order terdapat dua pilihan jenis pengiriman yaitu pengiriman semalam atau pengiriman biasa.
Selanjutnya diakhiri oleh aksi tutup order.



Gambar 11.1 Diagram aktivitas sistem order

Diagram aktivitas dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

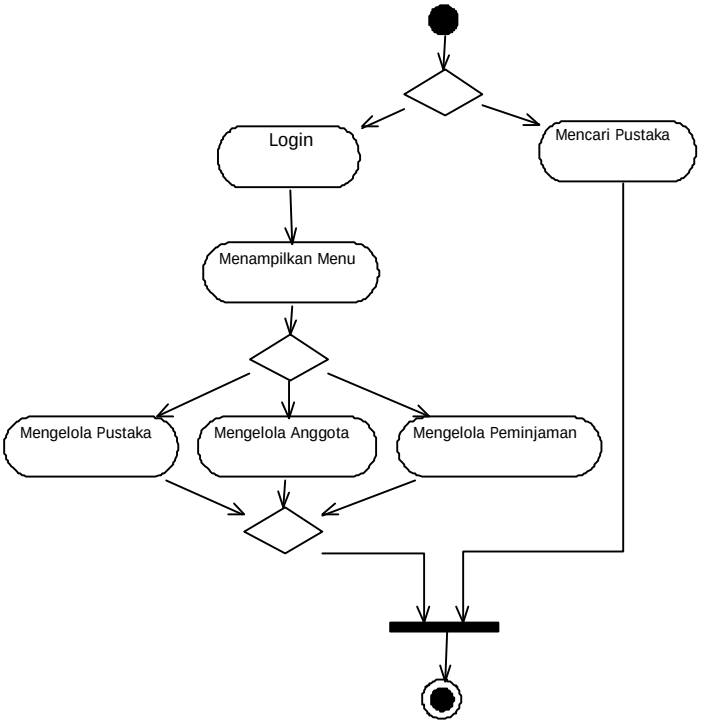
Contoh kasus di atas dengan *swimlane* :



Gambar 11.2 Diagram aktivitas sistem order dengan *Swimlane*

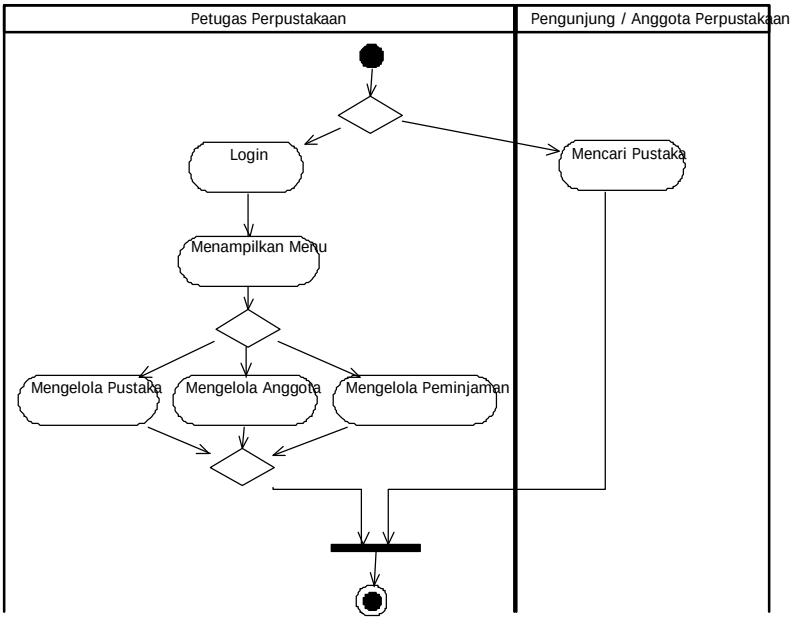
9.3 *Studi Kasus Diagram Aktivitas*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram aktivitas dari sistem informasi manajemen perpustakaan:



Gambar 15 Diagram Interaksi Studi Kasus

Jika digambar dengan menggunakan swimlane maka akan menjadi sebagai berikut:



Gambar 16 Diagram Aktivitas dengan *Swimlane*

PDF-XChange
Click to buy NOW!
www.docu-track.com

Kuis Benar Salah

1. Diagram aktivitas mirip dengan diagram alir sistem (system *flowchart*).
2. Diagram aktivitas menunjukkan tahapan, pengambilan keputusan dan percabangan.
3. Diagram aktivitas tidak dapat menggambarkan kegiatan paralel.
4. Pada diagram aktivitas dan diagram status diawali oleh status awal yang simbolnya sama.
5. Terdapat aliran ke kegiatan diri sendiri pada diagram aktivitas.
6. Diagram aktivitas tidak dapat menunjukkan pelaku yang bertanggung jawab mengerjakan aksi-aksi tertentu.
7. Tanda status akhir di diagram aktivitas mungkin tidak ada.
8. Diagram aktivitas dengan *swimlane* tidak menunjukkan aktivitasnya.
9. Di akhir suatu kegiata yang bersamaan harus dilakukan penggabungan.
10. Pesan pada diagram aktivitas tidak terlihat.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Simbol lingkaran solid diarsir adalah : _____

   A. awal
   B. akhir
   C. status
   D. aksi
   E. transisi

2. Simbol oval adalah : _____

   A. awal
   B. akhir
   C. koondisi
   D. event
   E. aksi

3. Simbol belahketupat adalah : _____

   A. awal
   B. akhir
   C. keputusan
   D. Aksi paralel
   E. aksi

4. Kegiatan pilihan dalam diagram aktivitas ditunjukkan dengan membuat simbol _____ di awal dan di akhir keputusan.

   A. Status
   B. fork
   C. Keputusan
   D. Kegiatan
   E. obyek

5. Berikut adalah komponen dari diagram aktivitas, kecuali :

   A. Titik awal
   B. Titik akhir
   C. *fork*
   D. aksi
   E. aktor

4. Objek yang bergerak dari satu status ke status yang lain adalah : _____

   A. awal
   D. event

B. akhir
C. status
E. transisi

5. Berikut adalah komponen dari diagram status, kecuali :

A. Titik awal
B. Titik akhir
C. *fork*
D. aksi
E. aktor

6. Diagram aktivitas digunakan untuk melengkapi diagram : ____

A. Sekuen
B. Status
C. Kolaborasi
D. A, B dan C benar
E. Tidak ada jawab

7. Akhir dari kegiatan paralel harus digabung dengan simbol : _____

A. Status akhir
B. *Join*
C. *Fork*
D. keputusan
E. aktivitas

8. Awal kegiatan paralel pada diagram aktivitas adalah : _____

A. Status awal
B. *Join*
C. *Fork*
D. keputusan
E. Status akhir

9. Diagram aktivitas termasuk diagram dinamis seperti diagram di bawah ini, kecuali : _____

A. Diagram kelas
B. Diagram kolaborasi
C. Diagram status
D. Diagram sekuen
E. Salam semua

10. Nama objek yang bertanggung jawab dapat dimunculkan pada diagram aktivitas, hal ini disebut : _____

A. Aksi
B. percabangan
C. swimlane
D. *fork*
E. join

Latihan

1. *Basic*
    a. Apakah yang dimaksud dengan *activity diagram*?
    b. Apa kegunaan *activity diagram*?
    c. Apakah yang dimaksud dengan *swimlane* pada *activity diagram*?
2. *Advanced*
    a. Kapan (dalam kondisi seperti apa) *activity diagram* sebaiknya digunakan?
    b. Buatlah *activity diagram* untuk sistem informasi apotek!

10 Diagram Komponen

Overview

Bab ini berisi mengenai cara untuk membuat diagram komponen pada sistem informasi. Diagram komponen digunakan untuk menggambarkan komponen-komponen yang ada pada sistem informasi. Diagram komponen digunakan untuk memodelkan aspek fisik suatu sistem. Aspek fisik ini berupa modul-modul yang berisikan *code*, baik *library* maupun *executable, file* atau dokumen yang ada di dalam *node*. Aspek fisik inilah yang dikatakan komponen dalam UML.
Umumnya komponen yang terbentuk dari beberapa *class* dan/atau *package*, atau juga dapat dari komponen-komponen yang lebih kecil.

Tujuan

1. Mahasiswa memahami tujuan penggunaan diagram komponen.
2. Mahasiswa mengetahui bagaimana cara untuk membuat diagram komponen.
3. Mahasiswa mampu merancang diagram komponen untuk sebuah sistem informasi yang sederhana.

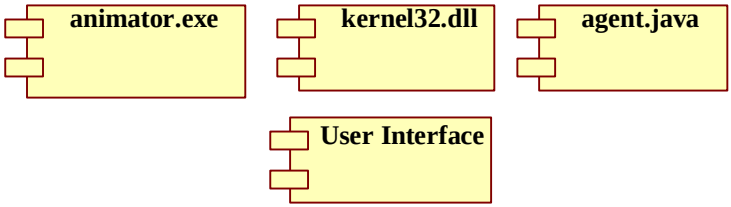
10.1 *Pengertian Diagram Komponen*

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.

Pengertian komponen sendiri dalam UML adalah hal-hal fisik dari sistem yang akan dimodelkan dan ada ketika sistem dieksekusi. Diantara contoh komponen dasar pada sebuah Sistem yaitu :

- Komponen *user interface* yang menangani tampilan
- Komponen *bussiness processing* yang menangani fungsi-fungsi proses bisnis
- Komponen *data* yang menangani manipulasi data
- Komponen *security* yang menangani keamanan sistem

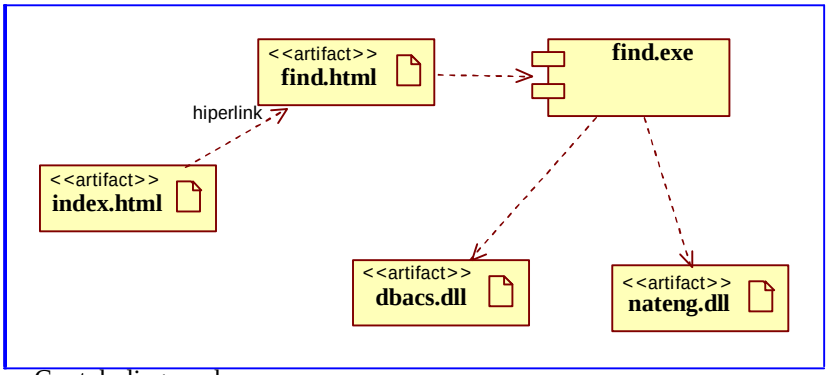
Contoh lain komponen dalam perangkat lunak yaitu operating sistem, bahasa pemrograman, obyek-obyek library, file executable, COM+. Termsuk juga dapat dimodelkan sebagai komponen adalah tabel, file (*source code*) dan dokumen.



Ilustrasi aspek fisik sistem yang dapat dimodelkan sebagai komponen dalam UML

Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

- *source code* program perangkat lunak
- komponen *executable* yang dilepas ke *user*



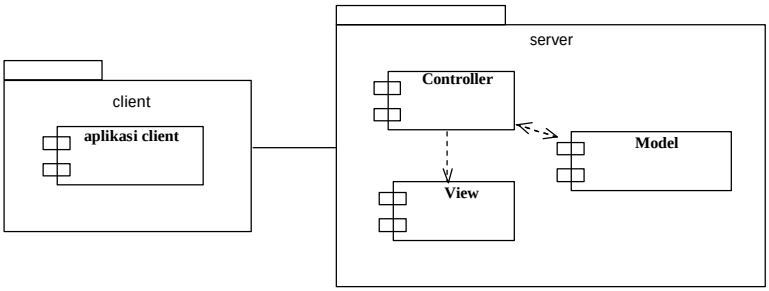
Contoh diagram komponen

Ket : - page = index.html, find.html
- executable file = find.exe
- library = dbacs.dll, nateng.dll
- komponen = semua hal diatas termasuk komponen

basis data secara fisik

sistem yang harus beradaptasi dengan sistem lain

framework sistem, *framework* pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi, contohnya seperti Struts dari Apache yang menggunakan prinsip desain Model-View-Controller (MVC) dimana *source code* program dikelompokkan berdasarkan fungsinya seperti pada gambar berikut:



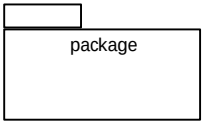
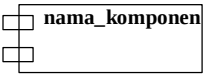
Gambar 17 Ilustrasi *Framework*

Dimana *control er* berisi *source code* yang menangani *request* dan validasi, *model* berisi *source code* yang menangani manipulasi data dan *business logic*, dan *view* berisi *source code* yang menangani tampilan.

Diagram komponen mengandung komponen, *interface* dan hubungan (*relationship)* yang mengandung kebergantungan antar komponen. Diagram komponen ini digunakan pada saat ingin memecah sistem menjadi komponen-komponen dan ingin menampilkan hubungan-hubungan mereka dengan antarmuka atau pemecahan komponen menjadi struktur yang lebih rendah.

Secara umum dapat dikatakan bahwa diagram komponen digunakan untuk menjelaskan kebergantungan antar beragam komponen-komponen *software* seperti misalnya kebergantungan antara *file-file executable* dengan *file-file* sumbernya (*sourcefile*) dll.

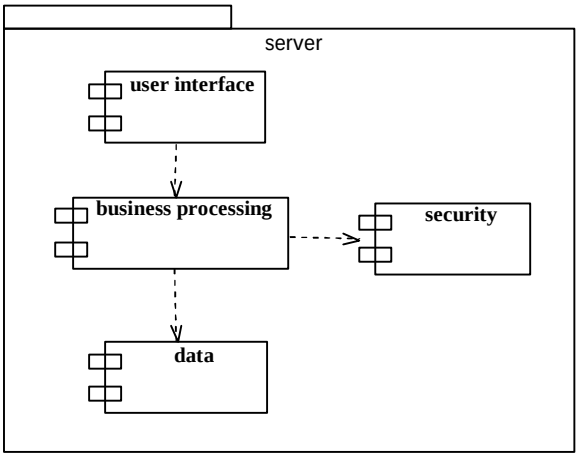
Berikut adalah simbol-simbol yang ada pada diagram komponen:

Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih komponen
Komponen 	Komponen sistem
Kebergantungan / *dependency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
Antarmuka / *interface*	sama dengan konsep *interface* pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen

Simbol	Deskripsi
nama_interface	agar tidak mengakses langsung komponen
Link	relasi antar komponen
	Dokumen, dapat berupa file, library

10.2 *Studi Kasus Diagram Komponen*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram komponen dari sistem informasi manajemen perpustakaan:



Gambar 18 Diagram Komponen Studi Kasus

Kuis Benar Salah

1. Sebuah komponen tidak menggambarkan hal-hal fisik dari sistem.
2. Komponen disimbolkan dengan kotak persegi panjang dengan yang memiliki tab.
3. Diagram komponen merupakan pandangan statis implementasi.
4. Tidak diperlukan hubungan kebergantungan antar komponen dalam diagram komponen.
5. *Source code* dan *executable files* termasuk komponen.
6. Yang termasuk *runtime component* yaitu executable files, DLL files dan task.
7. Interface dalam diagram komponen dilambangkan dengan belah ketupat.
8. Diagram komponen digunakan untuk untuk menjelaskan kebergantungan komponen dalam sistem.
9. *Depedency* dilambangkan dengan garis lurus tanpa putus-putus.
10. Framework sistem Struts menggunakan prinsip desain Model-View-Controller (MVC).

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Diantara tujuan diagram komponen kecuali :
   A. Visualisasi komponen
   B. Penjelasan organisasi komponen
   C. Membangun file yang dapat di eksekusi
   D. A, B dan C benar
   E. Tidak ada jawab

2 Aspek fisik sistem yang dapat dimodelkan menjadi komponen, kecuali _____
   A. COM+
   B. File library
   C. Source code
   D. Aturan akses ke login
   E. Bukan salah satu

3. Dalam konsep dasar UML, diagram komponen termasuk pada : _____
   A. *Static view*
   B. *Implementation View*
   C. *Deployment View*
   D. *Interaction view*
   E. *Activity view*

4. Antar muka agar hubungan tidak langsung mengakses ke komponen :
   A. Link
   B. Package
   C. Interface
   D. *Depedency*
   E. Dokumen

5. Simbol lingkaran bulat yaitu : _____
   A. Link
   B. Package
   C. Interface
   D. *Depedency*
   E. Dokumen

6. Yang termasuk *runtime component*, kecuali :
   A. Task
   B. DLL files
   D. File exe
   E. A, C, D benar

C. Swimline

7. Simbol kebergantungan antar komponen : _____
   A. Garis solid
   B. Belah ketupat
   C. *Join*
   D. Garis putus-putus
   E. Kubus

8. Simbol dokumen atau artifak yaitu :
   A. Garis solid
   B. Belah ketupat
   C. Gambar kertas
   D. Kubus
   E. Lingkaran

9. Yang termasuk file libarary yaitu :
   A. Find.exe
   B. Login.html
   C. Kernel32.dll
   D. A,B,C Benar
   E. Tidak ada yang benar

10. Pembungkus komponen-komponen dalam diagram komponen : _____
    A. Aksi
    B. Package
    C. swimlane
    D. Interface
    E. Join

Latihan

1. *Basic*
    a. Apakah yang dimaksud dengan *component diagram*?
    b. Apa kegunaan *component diagram*?
    c. Apakah yang dimaksud dengan MVC dan apa fungsinya?
2. *Advanced*
    a. Kapan (dalam kondisi seperti apa) *component diagram* sebaiknya digunakan?
    b. Buatlah *component diagram* untuk sistem informasi apotek!

11 Diagram *Deployment*

Overview

Bab ini berisi mengenai cara membuat diagram *deployment*. Diagram *deployment* digunakan untuk menggambarkan konfigurasi komponen pada saat instalasi sistem informasi. Diagram *deployment* menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen *hardware* tersebut.

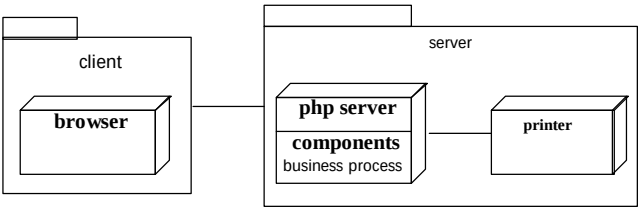
Tujuan

1. Mahasiswa memahami tujuan penggunaan diagram *deployment*.
2. Mahasiswa mengetahui cara untuk membuat diagram *deployment* pada suatu sistem informasi.
3. Mahasiswa mampu merancang diagram *deployment* untuk sebuah sistem informasi yang sederhana.

11.1 *Pengertian Diagram* Deployment

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut:

- sistem tambahan (*embedded system*) yang menggambarkan rancangan *device*, *node*, dan *hardware.*
- sistem *client/server* misalnya seperti gambar berikut:



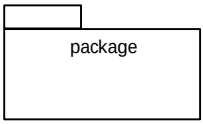
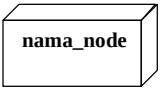
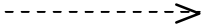
Gambar 19 Diagram *Deployment* Sistem *Client / Server*

- sistem terdistribusi murni
- rekayasa ulang aplikasi

Diagram *deployment* mewakili pandangan pengembangan sistem sehingga hanya akan ada satu diagram *deployment* untuk satu sistem. Diagram *deployment* terdiri dari node dan node merupakan perangkat keras fisik yang digunakan untuk menyebarkan aplikasi. Diagram *deployment* banyak digunakan oleh *system engineer*.
Tiap node pada diagram deployment mewakili satu unit komputasi sistem yang dalam banyak hal merupakan bagian dari perangkat keras.

Diagram *deployment* umumnya memiliki node dan hubungan kebergantungan. Memungkinkan juga dalam diagram *deployment* terdapat komponen.

Berikut adalah simbol-simbol yang ada pada diagram *deployment*:

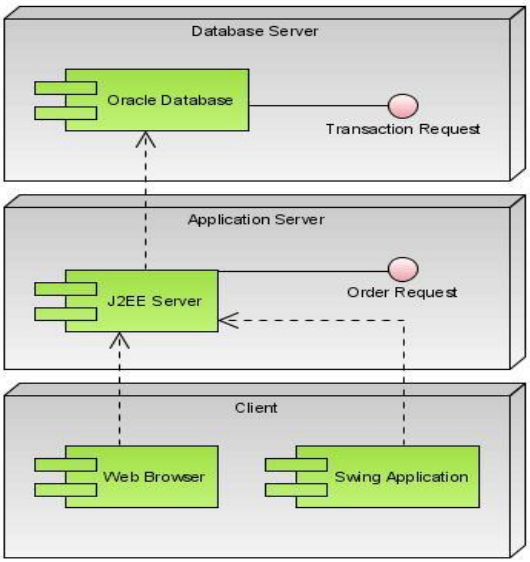
Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih *node*
Node 	biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / *dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
Link	relasi antar *node*

11.2 *Cara menentukan diagram* deployment *arsitektur sistem.*

Diagram *deployment* juga dapat dibuat untuk **mencari arsitektur** yang tertanam sistem, menunjukkan bagaimana komponen perangkat keras dan perangkat lunak saling bekerja sama.

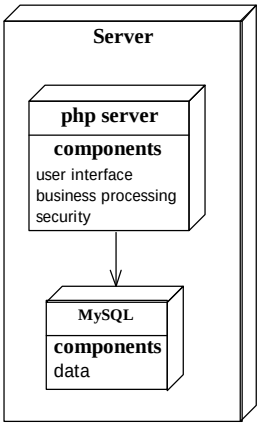
Ada beberapa langkah untuk menentukan diagram *deployment* :

1. Mengidentifikasi lingkup model, yaitu apakah dari sistem dalam organisasi hanya menggunakan satu aplikasi yang terintegrasi.
2. Mempertimbangkan hal-hal teknis yang dasar, diantaranya mengenai :
   - Apakah sistem yang ada perlu ada interaksi sehingga harus diintegrasikan dengan perangkat keras?
   - Bagian mana dan bagaimana jenis interaksi dan hubungan yang akan dilakukan? Misalnya menggunakan internet, sharing file dll.
   - Sistem operasi, perangkat komunikasi dan jenis protokol apa yang akan digunakan
   - Apakah perangkat lunak dan perangkat keras akan langsung berhubungan dengan pengguna
   - Bagaimana sistem keamanan sistemnya
3. Mengidentifikasikan arsitektur jaringan (distribusi), misalnya apakah akan menggunakan aplikasi server terpusat atau terdistribusi sehingga tingkat distribusi aplikasi *two tier* atau *three tier*?
4. Mengidentifikasikan node dan koneksi, yaitu bagaimana antar node dan komponen akan berhubungan dan bergantung
5. Mendistribusikan perangkat lunak ke node



11.3 *Studi Kasus Diagram Deployment*

Studi kasus diambil dari sistem informasi manajemen perpustakaan seperti pada bab-bab sebelumnya. Berikut adalah diagram *deployment* dari sistem informasi manajemen perpustakaan:



Gambar 20 Diagram *Deployment* Studi Kasus

Aplikasi di-*deploy* pada sebuah komputer server dimana di dalamnya sudah terdapat php server dan MySQL sebagai DBMS.

Kuis Benar Salah

1. Diagram *deployment* menunjukan layout sistem secara fisik yang menunjukan bagian-bagian software yang berjalan pada bagian-bagian hardware.
2. Dalam satu aplikasi dimungkinkan banyak diagram *deployment*.
3. Diagram *deployment* banyak digunakan oleh *system engineer*.
4. Node dilambangkan dengan lingkaran.
5. Didalam node dapat digambarkan komponen namun harus sama dengan yang ditampilkan dalam diagram komponen sebelumnya.
6. Node dapat dibungkus dalam satu *package*.
7. Antar node dapat direlasikan dalam kebergantungannya dengan simbol garis tebal tanpa putus-putus.
8. Diagram *deployment* dapat digunakan untuk memodelkan *embedded system.*
9. Konfigurasi jaringan dan perangkat keras tidak termasuk dalam diagram *deployment*.
10. Arsitektur sistem dapat digambarkan dengan diagram *deployment* ini.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Simbol kubus (*cube*) adalah : _____
   A. Link
   B. Package
   C. Node
   D. *Dependency*
   E. Bukan salah satu
2. Simbol garis anak panah putus-putus adalah : _____
   A. Link
   B. Package
   C. Node
   D. *Dependency*
   E. Bukan salah satu
3. Yang dapat dimodelkan dalam diagram *deployment*, kecuali :
   A. *Embedded system*
   B. Client-server
   C. Sistem terdistribusi
   D. Rekayasa ulang aplikasi
   E. Hubungan antar komponen
4. Pemakai diagram *deployment* umumnya adalah : _____
   A. Database admin
   B. Programmer
   C. System engineer
   D. Project Leader
   E. Operator
5. Berikut adalah bagian dari diagram *deployment*, kecuali :
   A. Node
   B. Package
   C. Swimline
   D. Komponen
   E. Link
6. Diantara perangkat keras yang dapat dimodelkan dalam diagram *deployment* kecuali :
   A. Monitor
   B. Printer
   C. Mouse
   D. A, B dan C benar
   E. Tidak ada jawab

7. Hubungan antar node dengan simbol : _____
   A. Garis solid
   B. *Join*
   C. *Fork*
   D. Garis putus-putus
   E. Kubus
8. Yang perlu diidentifikasi dalam menentukan arsitektur menggunakan diagram *deployment*, kecuali :
   A. Lingkup model
   B. Distribusi software
   C. Node dan koneksinya
   D. Anggaran biaya
   E. Arsitektur jaringan
9. Komponen-komponen yang mungkin ada dalam sebuah node Main Server, kecuali
   A. ActiveX Control
   B. Java Bean
   C. Web Server
   D. A,B,C Benar
   E. Browser
10. Yang termasuk dalam satu area utama dalam konsep dasar UML dengan diagram *deployment* yaitu _____
    A. Diagram komponen
    B. Diagram sekuen
    C. Diagram aktivitas
    D. Diagram Status
    E. Salah semua

Latihan

1. *Basic*
   a. Apakah yang dimaksud dengan *deployment diagram*?
   b. Apa kegunaan *deployment diagram*?
   c. Apakah perbedaan antara *component diagram* dengan *deployment diagram*?
2. *Advanced*
   a. Kapan (dalam kondisi seperti apa) *deployment diagram* sebaiknya digunakan?
   b. Buatlah *deployment diagram* untuk sistem informasi apotek!

12 Kohesi dan Kopling

Overview

Bab ini berisi mengenai jenis pengukuran untuk mendapatkan desain berorientasi obyek memiliki teknik yang baik. Akan dijelaskan tentang kohesi dan kopling sebagai jenis pengkuran fungsi independen yang merupakan kunci kualitas program.

Tujuan

1. Mahasiswa memahami pengertian kohesi.
2. Mahasiswa mengetahui pengertian kopling.
3. Mahasiswa mampu mengetahui teknik mendesain OO yang baik.

12.1 *Pendahuluan*

Functional Independence merupakan kunci perancangan yang baik dan kunci kualitas program.

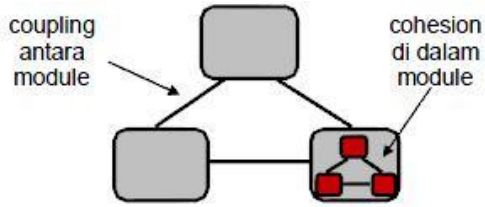
- Merupakan hasil pertumbuhan langsung konsep abstraksi dan *information hiding*.
- Memiliki subfungsi yang spesifik dan antarmuka yang sederhana apabila dipandang dari bagian lain dalam struktur program.

Keuntungan Modul yang independen:

- mudah membagi dalam tim,
- mudah diubah,
- perambatan kesalahan berkurang, dan
- *reusable* bertambah.

Independensi fungsional ini diukur dengan menggunakan dua kriteria yaitu :

- Kohesi (keterpautan)
- Kopling (keterkaitan)



Hubungan antara kohesi dan koping

12.2 *Kohesi*

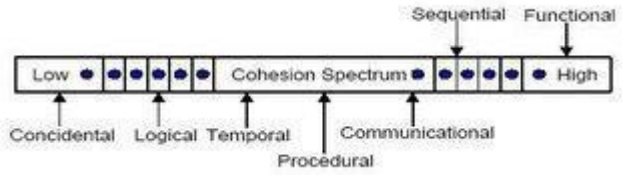
Kohesi (*Cohesion*) adalah ukuran keterpaduan dimana hubungan antara elemen-elemen dalam suatu modul jelas dan terstruktur. Kriteria ini untuk mengukur seberapa independen sebuah program dari pada prosesnya sendiri. Sehingga sebuah program yang kohesi akan memiliki data dan logika yang diperlukan untuk melengkapi aplikasinya tanpa secara langsung terpengaruh oleh program lain.

Program yang kohesif dapat dimisalkan dengan bagian mobil yang dapat ditukar-tukarkan. Jika sebuah mobil memiliki ukuran ban 14 inchi maka semua

jenis dan merek ban asalkan ukurannya 14 inchi dapat dipasangkan pada mobil ini. Jadi ban mobil 14 inchi ini, tidak dikhususkan pada satu atau beberapa jenis mobil tertentu, tetapi merupakan komponen yang kohesif untuk banyak tipe mobil.

Jenis-jenis kohesi yaitu :

1. *Coincidental cohesion* (paling rendah):
   Jika modul/kelas terdiri dari beberapa fungsi tetapi tidak terdapat hubungan yang berarti antara elemen-elemen dari modul tersebut. Suatu kejadian dimana secara kebetulan saja bahwa elemen-elemen berada dalam tempat yang sama.
2. *Logical y cohesion* :
   Jika terdiri dari beberapa fungsi yang mempunyai tugas serupa atau melakukan fungsi-fungsi yang masuk dalam kelas logika yang sama.
3. *Temporal cohesion* :
   Jika sebuah modul berisi sejumlah tugas yang dihubungkan dengan segala yang harus dieksekusi di dalam waktu yang bersamaan.
4. *Procedural cohesion* :
   Jika pemrosesan elemen-elemen dari suatu modul dihubungkan dan harus dieksekusi dalam urutan spesifik.
5. *Communication cohesion* :
   Jika pemrosesan elemen-elemen dikonsentrasikan pada satu area dari suatu struktur data.
6. *Sequential cohesion.*
   Jika keluaran dari suatu elemen merupakan masukan untuk elemen yang lain secara berurutan. Modul terdiri dari beberapa fungsi dimana elemen dalam modul bertindak sebagai suatu himpunan urutan aksi-aksi atau fungsi-fungsi sangat berikatan.
7. *Functional cohesion* (paling tinggi)
   Bila seluruh elemen dari modul terkait hanya melakukan suatu fungsi tunggal yang terdefinisi dengan baik, tanpa tergantung pada implementasi modul-modul yang lain.



Kohesi yang baik adalah yang *high cohesion*.

Kohesi sebuah sistem akan menurun jika :

- Metode dari suatu kelas hanya sedikit yang sama
  Metode mengerjakan aktivitas yang banyak sehingga tidak saling berkaitan.

Kekurangan dari *low cohesion* yaitu :

- Kesulitan dalam memahami modul/kelas
  Kesulitan dalam memelihara sistem sebab perubahan lojik dalam modul/kelas akan mempengaruhi banyak modul sehingga jika satu modul/kelas berubah maka harus mengubah modul/kelas lain
- Kesulitan untuk menggunakan ulang modul/kelas, karena tidak ada modul/kelas lain yang memerlukan.

12.3 *Kopling*

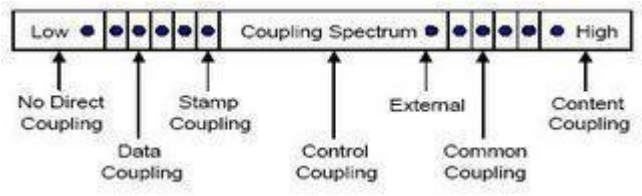
Kopling dapat diartikan bagaimana sebuah aplikasi atau modul atau kelas saling berhubungan dan bergantung. Modul yang tidak tergantung kepada modul lain dikatakan dengan memiliki kopling yang rendah.

Jenis-jenis coupling yaitu :

1. *Data coupling*
   Dua buah modul dari sistem mempunyai *data coupling* jika komunikasi dari modul-modul ini dilakukan lewat suatu data. Data dapat berupa sebuah item data tunggal atau elemen dari suatu larik (array).
2. *Stamp coupling*
   Dua buah modul dikatakan *stamp coupling* jika kedua modul ini berkomunikasi lewat suatu kelompok item data. Kelompok item data dapat berupa suatu record yang terdiri dari beberapa field atau larik yang terdiri dari beberapa elemen.
3. *Control coupling*
   Control coupling ditandai dengan adanya alur kendali antara modul. Modul satu mengendalikan aliran data di modul yang lain, kendali dilakukan melalui flag yang mengontrol logika intern modul yang lain.
4. *External coupling*
   Tingkat *coupling* yang terjadi bila modul-modul terikat pada lingkungan luar (external) dari perangkat lunak.
5. *Common coupling*
   Modul-modul dikatakan *common coupling* jika modul-modul tersebut menggunakan data yang disimpan di area memori yang sama.

6. *Content coupling*

Content coupling terjadi jika suatu modul menggunakan data atau mengendalikan informasi dari modul yang lain tanpa berhubungan lewat suatu parameter. *Content coupling* dapat juga terjadi jika percabangan dilakukan ke tengah-tengah suatu modul yang lain.



12.4 *Teknik desain* object oriented *yang baik*

Dalam hal OOP sebisa mungkin keterikatan kelas dengan kelas lain itu diminimalkan alias *loose coupling*. Sementara kohesi berbicara tentang fungsi dalam konteks dimana semakin kedalam tingkat hirarki kelas-kelas itu spesifik fungsinya. Atau juga membagi kelas-kelas sedemikian rupa sehingga setiap kelas menjalankan fungsi spesifik (atau yang mempunyai keterkaikan) dan tidak terkumpul dalam sebuah kelas besar. Untuk kohesi yang diharapkan adalah *high cohession.*

Reusable adalah kunci pokok dalam pengembangan perangkat lunak, tema inilah yang mengilhami perancangan modul program dan perkembangan paradigma pengembangan perangkat lunak secara umum.
Reusable bisa diperoleh bila menerapkan *information hiding*.

Ciri Modul yang baik:

- *High cohesion* (*functional cohesion*): modul hanya melakukan satu tugas dan memerlukan sedikit interaksi dengan modul lain dalam satu program.

- *Low coupling*: modul memiliki kopling antar modul yang lemah atau sebebas mungkin dengan modul yang lain (independen). Kopling tergantung pada kompleksitas antarmuka modul.

Kuis Benar Salah

1. Independensi fungsional merupakan turunan dari konsep abstraksi dan *information hiding*
2. Keuntungan dari modul/kelas yang independen yaitu kemampuan modul untuk dipakai kembali oleh modul lain.
3. Modul yang dirancang agar informasi yang berada di dalam modul tidak dapat diakses oleh oleh modul lain yang tidak memerlukan informasi tersebut merupakan pengertian *information hiding*.
4. Kopling merupakan ukuran keterpaduan antar elemen dalam satu modul/kelas.
5. Kohesi yang paling rendah yaitu jika tidak ada hubungan yang kuat antar elemen dalam modul tersebut.
6. Modul yang tidak bergantung kepada modul lain dikatakan memiliki kopling yang tinggi.
7. Kopling yang paling rendah yaitu No Direct Coupling.
8. Hal yang paling mendasar dalam pengukuran OO hanya pada kohesi saja.
9. Teknik desain OO yang baik adalah kopling tinggi, kohesi tinggi.
10. Kopling yang rendah maksudnya hubungan antar modul/kelas yang tidak terikat dan independen.

Pilihan Ganda

Petunjuk: Pilihlah jawaban yang paling tepat!

1. Diantara keuntungan modul yang independen, kecuali
   A. Mudah pembagian tim
   B. *Reusable* bertambah
   C. Mudah diubah
   D. Perambatan masalah berkurang
   E. Tergantung dengan modul lain
2. Jika modul memiliki aksi/fungsi yang mempunyai tugas dalam kelas logika yang sama
   A. *Concidental cohesion*
   B. *Logical cohesion*
   C. *Communicational Cohesion*
   D. *Functional cohesion*
   E. *Sequential cohesion*
3. Dalam satu modul hanya terdapat satu aksi tunggal tanpa tergantung modul-modul lain.
   A. *Concidental cohesion*
   B. *Logical cohesion*
   C. *Communicational Cohesion*
   D. *Functional cohesion*
   E. *Sequential cohesion*
4. Jenis kohesi yang diharapkan sebagai salah satu teknik perancangan OOP adalah : _____
   A. *Concidental cohesion*
   B. *Logical cohesion*
   C. *Communicational Cohesion*
   D. *Functional cohesion*
   E. *Sequential cohesion*
5. Diantara jenis-jenis kopling kecuali :
   A. Data
   B. Stamp
   C. Content
   D. Control
   E. Procedural
6. Jenis kopling yang memiliki kualitas terbaik yaitu :
   A. Data
   B. Stamp
   D. Control
   E. Common

C. Content

7. Jenis kopling yang berisi informasi yang paling rumit, tidak jelas dan dengan jenis hubungan tidak dengan parameter yaitu :

_____

A. Data
B. Stamp
C. Content
D. Control
E. Common

Gunakan data ini untuk mengerjakan no 8-10.

1. Kohesi tinggi
2. Kohesi rendah
3. Kopling rendah
4. Kopling tinggi
5. tidak ada kopling

8. Teknik desain OO yang paling baik :

A. 1 dan 4
B. 1 dan 3
C. Tidak ada jawaban
D. 2 dan 3
E. 2 dan 4

9. Dua buah modul dari sistem mempunyai *data coupling* jika komunikasi dari modul-modul ini dilakukan lewat suatu data.

A. 1 saja
B. 1 dan 3
C. 3 saja
D. 5
E. Salah semua

10. Hubungan antar modul/kelas yang tidak terikat dan independen

A. 1
B. 2
C. 3
D. 4
E. 5

Latihan

Diberikan dua blok program yaitu A dan B sebagai berikut :

A
X=10
Y=20
Z=2(X+Y)

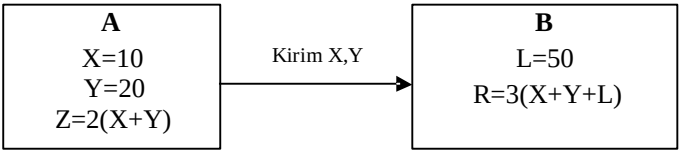
B
X=10
L=50
R=3(X+Y+L)

Kondisi awal :

Hubungan antara program A dan program B, adalah dari program A akan mengirimkan data variabel X ke program B. Analisa kondisi seperti ini, apa kekurangannya.

Kondisi akhir :

Hubungan antara program A dan program B, yaitu program A mengirimkan sekaligus dua variabel yaitu X dan Y ke program B, sehingga variabel tersebut tidak perlu di definisikan lagi di program B. Analisa apa keuntungan dari hal ini. Bagaimana Anda menjelaskan *low coupling* ? dan bagaimana pula program A menjadi lebih kohesi ?



Daftar Pustaka

Booch, Grady. 1998. Object-oriented analysis and design with applications 2nd edition. Addison Wesley.

Grady Booch, James Rumbaugh and Ivar Jacobson, The UML User‟s Guide, 1st Edition, Addison and Wesley, 1998.

Bowman, Kevin. 2004. System Analysis: A Beginner‟s Guide. Palgrave Macmillan.

Knudson, Joan dan Ira Bitz. 1991. Project Management. Amacom.

Langer, Arthur M. 2008. Analysis and Design of Information Systems 3rd edition. Springer.

Rumbaugh, James dkk. 1999. The Unified Modeling Language Reference Manual. Addison-Wesley.

Simon Bennet, Steve McRobb, Ray Farmer, Object Oriented Systems Analysis and Design Using UML 2nd, McGraw Hill, 2002.