Analisis dan Perancangan Berorientasi Objek

(Digunakan di lingkungan sendiri, sebagai buku ajar mata kuliah Analisis dan Perancangan Berorientasi Objek)

Fakultas Teknik dan Ilmu Komputer

Program Studi Sistem Informasi

Universitas Komputer Indonesia

1 Pertemuan 1

1.1 Pendahuluan

1.1.1 Ruang Lingkup Matakuliah

Analisis berorientasi objek (OOA) adalah proses menganalisis tugas, untuk mengembangkan sebuah model konseptual yang kemudian dapat digunakan untuk menyelesaikan tugas. Sebuah model OOA yang mempunyai ciri khas khusus akan menjelaskan perangkat lunak komputer yang dapat digunakan untuk memenuhi keperluan pengguna berdasarkan kriteria yang telah ditentukan. Selama fase analisis pemecahan masalah, analis mungkin mempertimbangkan pernyataan-persyaratan tertulis, dokumen visi formal atau wawancara dengan para pemangku kepentingan atau pihak berkepentingan lainnya. Tugas yang ditangani mungkin akan dibagi menjadi beberapa sub-tugas (atau domain), masing-masing mewakili bisnis yang berbeda, teknologi, atau daerah lain yang menarik. Setiap subtask akan dianalisis secara terpisah. Pelaksanaan kendala, (misalnya, konkurensi, distribusi, ketekunan, atau bagaimana sistem ini akan dibangun) tidak dipertimbangkan selama tahap analisis, melainkan, mereka ditangani selama desain berorientasi objek (OOD).

1.1.2 Sasaran

Selama perancangan berorientasi objek (OOD), pengembang mendapatkan masalah dalam pengimplementasian untuk model konseptual yang akan dihasilkan dalam analisis berorientasi objek. Kendala tersebut dapat mencakup tidak hanya pada masalah yang diberikan oleh arsitek yang dipilih, tetapi juga non-fungsional - teknologi atau lingkungan - masalah, seperti transaksi throughput, waktu respon, run-time platform, lingkungan pengembangan atau mereka yang melekat dalam bahasa pemrograman. Konsep dalam model analisis yang dipetakan ke kelas implementasi dan interface yang dihasilkan dalam solusi model domain, yaitu penjelasan rinci tentang bagaimana sistem ini akan dibangun.

1.1.3 Tujuan

Setelah menyelesaikan mata kuliah ini, diharapkan mahasiswa mampu memformulasikan suatu persoalan dan memberikan solusinya dalam bentuk program berorientasi object.

2 Pertemuan 2

2.1 Kerangka umum pembuatan sistem

Dalam membuat sebuah system pastilah terdapat beberapa tahap dalam pembuatan kerangka umum system informasi agar system yang kita buat dapat menhasilkan sebuah system yang mudah dipahami oleh semua orang yang memakai atau membutuhkannya. Ini adalah tahapan atau kerangka dalam pembuatan sebuah system informasi :

a. Perencanaan

Perencanaan adalah membuat semua rencana yang berkaitan dengan proyek sistem informasi. kalau kita ingin membangun rumah maka kita akan melakukan perencanaan bagaimana pondasinya , bagaimana struktur bangunannya, mau memakai material apa saja, apa warna dindingnya, tak ketinggalakan pula merencanakan anggaran budget yang harus kita keluarkan.

b. Analisa

Analisa adalah menganalisa workflow sistem informasi yang sedang berjalan dan mengindentifikasi apakah workflow telah efisien dan sesuai standar tertentu.

c. Desain

adalah membuat desain (*desgin*). Desain adalah langkah yang sangat penting dalam siklus SDLC karena langkah ini menentukan fondasi sistem informasi. kesalahan dalam desain dapat menimbulkan hambatan bahkan kegagalan proyek.

Ada 2 jenis desain yang dibuat di langkah ini, yaitu desain proses bisnis dan desain pemrograman.

1. Desain Proses Bisnis
2. Desain Pemrograman

Desain pemrograman dilakukan oleh Sistem Analis (SA) yaitu membuat desain yang diperlukan untuk pemrograman berdasarkan desain proses bisnis yang telah dibuat oleh BPA. desain ini akan menjadi pedoman bagi programmer untuk menulis *source code.*

Desain pemrograman meliputi :

a). Desain database
b). Desain Screen Layout
c). Desain Diagram Proses
d). Desain Report Layout

d. Pengembangan

Pekerjaan yang dilakukan di tahap pengembangan (development) adalah pemrograman. Pemrograman adalah pekerjaan menulis program komputer dengan bahasa pemrograman berdasarkan algoritma dan logika tertentu. orangnya disebut Programmer.

e. Testing

Testing adalah proses yang dibuat sedemikian rupa untuk mengidentifikasi ketidaksesuaian hasil sebuah sistem informasi dengan hasil yang diharapkan.

f. Implementasi

Implementasi adalah proses untuk menerapkan sistem informasi yang telah dibangun agar user menggunakannya menggantikan sistem informasi yang lama.

Proses Implementasi :

a. Memberitahu user
b. Melatih user
c. Memasang sistem (*install system*)
d. Entri/Konversi data
e. Siapkan user ID

g. Pengoperasian dan Pemeliharaan

Langkah Paling akhir adalah pengoperasian dan pemeliharaan. selama sistem informasi beroperasi, terdapat beberapa pekerjaa rutin yang perlu dilakukan terhadap sistem informasi, antara lain :

a. System Maintenance
b. Backup & Recovery
c. Data Archive

2.1.1 Mengenal pembuatan sistem informasi

Sistem Informasi Managemen merupakan proses komunikasi antara manusia dan mesin yang terpadu dan sistematis untuk menyajikan informasi guna mendukung fungsi operasi, aktifitas managemen dan pengambilan keputusan dalam suatu organisasi. Dalam managemen terdapat beberapa aktifitas yang dilakukan oleh manager, aktifitas tersebut pada dasarnya meliputi tiga hal yaitu (*planing*) bagaimana seorang manager merencanakan berbagai hal yang akan dilaksanakan dalam perusahaanya, (*organising*) bagaimana seorang manager mengorganisasikan perusahaan dan laryawanya, (*actuating*) proses dimana seorang

manager melakukan penguatan terhadap perusahaanya baik itu dengan cara memotivasi karyawan maupun mengevaluasi kembali kinerja perusahaanya, dan (*controling)* yaitu bagaimana manager mengendalikan perusahaan dan karyawanya agar tetap berjalan dengan baik.

Sedangkan dalam sistem informasi managemen terdapat informasi yang sangat berguna bagi manager untuk mengambil keputusan terbaik yang sesuai untuk perusahaanya. Dalam sistem informasi managemen terdapat informasi yang berasal dari hardware, software, brainware dan juga aturan-aturan yang saling terkait dan menjadi satu kesatuan informasi yang kemudian dijadikan sebagai input oleh perusahaan dan kemudian diproses oleh manager sehingga menghasilkan keputusan (*decision*) yang tepat bagi perusahaan. Hal-hal tersebut menunjukkan betapa pentingnya sistem informasi managemen bagi manager dalam mengambil keputusan, sebab informasi-informasi yang diperoleh mengenai kinerja perusahaan, bagaimana keberhasilan keputusan yang telah dibuat, apakah ada hal-hal lain yang belum terlaksana, apakah keputusan tersebut dapat berlaku jangka panjang dan lain sebagainya, oleh karena itu pada dasarnya manager dan sistem informasi managemen merupakan satu paket yang tidak dapat dipisahkan.

Terkait dengan keputusan dalam managemen, seperti yang disebutkan oleh Akhmad Subkhi dan Mohammad Jauhar dalam bukunya yang berjudul "Pengantar Teori dan Perilaku Organisasi", terdapat tiga jenis keputusan antara lain:

1. Keputusan terprogram yaitu keputusan yang dibuat untuk menangani situasi/masalah yang cukup sering terjadi, sehingga pembuat keputusan dapat membuat aturan-aturan pembuatan keputusan untuk diterapkan di masa depan. Misalnya keputusan untuk memesan persediaan ketika persediaan berada pada level tertentu.
2. Keputusan tidak terprogram yaitu keputusan yang dibuat dalam menanggapi situasi yang unik, tidak familier dan tidak terstruktur, serta menimbulkan konsekuensi-konsekuensi penting bagi organisasi. Banyak keputusan tidak terprogram melibatkan perencanaan strategis, karena ketidakpastianya

begitu besar dan keputusan merupakan hal yang sangat kompleks.

3. Keputusan setengah terprogram yaitu keputusan yang sebagian dapat diprogram, sebagian berulang-ulang dan rutin dan sebagian tidak terstruktur. Keputusan ini bersifat rumit dan membutuhkan perhitungan-perhitungan serta analisis yang terperinci.

Dilihat dari aspek individu sistem informasi managemen juga sangat bermanfaat karena, pada dasarnya setiap individu juga merupakan manager bagi kehidupanya sendiri. Informasi yang dia peroleh berdasarkan pencatatan pengeluaranya sehari-hari dan seberapa besar penghasilanya, kemudian akan diproses dan selanjutnya digunakan untuk menentukan keputusan dan bagaimana tindakan selanjutnya sehingga dia dapat mencukupi kebutuhanya saat ini dan dimasa mendatang. Oleh karena itu dalam mengatur kehidupan sehari-hari terutama masalah keuangan diperlukan pencatatan penghasilan dan pengeluaran yang dilakukan, baik itu selama periode perbulan, maupun pertahun. Sehingga dengan adanya informasi yang akurat dan kombinasi antara informasi dan proses pembuatan keputusanya, maka secara otomatis setiap individu dapat mengambil keputusan dengan lebih bijaksana.

Sehingga akan terdapat perbedaan diantara orang-orang yang menerapkan kegunaan sistem informasi managemen dalam keseharianya dengan orang sama sekali tidak pernah menerapkan sistem informasi managemen dalam kehidupanya. Misalkan saja dalam masalah yang sering dialami masyarakat indonesia yang cenderung memiliki sifat konsumtif. Itu bisa saja terjadi karena mereka tidak menerapkan managemen yang baik dalam keuangan mereka. Karena dengan buruknya kondisi managemen keuangan seseorang maka akan memperbesar tingkat pengeluaran daripada pendapatan. Maka melihat hal tersebut akan lebih baik jika dilakukan pencataatan penghasilan dan pengeluaran sehari-hari sebagai informasi (input) yang kemudian dapat diproses menjadi keputusan (output) yang tepat sesuai dengan kebutuhan hidupnya dalam jangka pendek (konsumsi) maupun jangka panjang (menabung/investasi).

2.1.2 Pendekatan dalam pembuatan sistem informasi

Proses pemecahan masalah secara sistematis bermulai dari John dewey, seorang profesor filosofi dari colombia university. Ia mengidenfikasikan tiga seri penelitian yang terlibat dalam memecahkan suatu kontroversi secara memadai.

1. Mengenali kontroversi
2. Menimbang klaim alternative
3. Membentuk penilaian

Serangkaian langkah pemecahan masalah yang memastikan bahwa maslah itu pertama-tama dipahami ,solusi alternatif dipertimbangkan, dan solusi yang dipilih bekerja.

Langkah-langkahnya adalah sbb:

a. Usaha persiapan = mempersiapkan manajer untuk memecahkan masalah dengan menyediakan orientasi sistem.

b. Usaha definisi = mencakup mengidentifikasi masalah untuk dipecahkan dan kemudian memahaminya.

c. Usaha solusi = mencakup mengidentifikasi berbagai solusi alternatif, mengevaluasinya, memilih satu yang tampak terbaik, menerapkan solusi itu dan membuat menindaklanjuti untuk menyakinkan bahwa masalah itu terpecahkan.

2.1.3 Karakteristik pembeda pendekatan objek dengan terstruktur

Pada pendekatan terstruktur merupakan metode yang pendekatannya pada proses, karena metode ini mencoba melihat system dari sudut pandang logical dan juga melihat data sebagai sumber proses. Di dalam penggambaran datanya, metode ini menggunakan Data Flow Diagram (DFD), Normalisasi, Entitas Relationship Diagram (ERD), dan lainnya.

Selain itu perbedaan yang paling mendasar dari pendekatan terstruktur dan objek adalah pada metode berorientasi fungsi atau aliran data (DFD), dekomposisi permasalahan dilakukan berdasarkan fungsi atau proses secara hirarki, mulai dari konteks sampai proses-proses yang paling kecil, sementara pada

pendekatan objek, dekomposisi permasalahan dilakukan berdasarkan objek-objek yang ada dalam system.

Untuk pendekatan objek, dalam melakukan pemecahan suatu masalah tidak dilihat bagaimana cara menyelesaikan suatu masalah tersebut tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Pendekatan ini memungkinkan pembangunan software dengan cepat, sehingga dapat segera masuk ke pasaran dan kompetitif. system yang dihasilkan sangat fleksibel dan mudah dalam pemeliharaan. Sedangkan untuk pengembangan terstruktur, menggunakan prosedur/tata cara yang teratur untuk mengoperasikan data struktur. Pendekatan ini mudah dimengerti oleh pengguna atau programmer, relative simple dan mudah dimengerti, berorientasi pada proses, sehingga mengabaikan kebutuhan nonfungsional.

3 Pertemuan 3

3.1 Pengenalan pendekatan objek

Pendekatan Objek merupakan paradigm pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigm ini dibungkus dalam kelas-kelas atau objek-objek, dimana setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Pendekatan objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan system.

Pendekatan Objek memiliki beberapa keuntungan, antara lain:

a. Maintenance. Program lebih mudah dibaca dan dipahami.
b. Pengubahan program (berupa penambahan ataupun penghapusan fitur tertentu). Perubahan ini antara lain menyangkut penambahan dan penghapusan dalam suatu database program misalnya.
c. Dapat digunakannya objek-objek sesering yang diinginkan.

Pendekatan Objek memiliki beberapa karakteristik atau sifat yaitu:

a. Abstraksi, yaitu prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan

mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

b. Enkapsulasi, yaitu pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek.
c. Pewarisan (Inheritance), yaitu mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.
d. Reusability, yaitu pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.
e. Generalisasi dan Spesialisasi, yaitu menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.
f. Komunikasi Antar Objek, yaitu dilakukan lewat pesan yang dikirim dari satu objek ke objek lainnya.
g. Polymorphism, yaitu kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama, sehingga menghemat baris program.

3.1.1 Kelahiran dan perkembangan pendekatan berorientasi objek

Gagasan orientasi objek sebenarnya dikembangkan sejak tahun 1965-an , perangkat pengembangnya lahir 1967-an,1990 mulai kajian secara insentif,1999 UML menjadilan Notasi bersama untuk pengembangan orientasi objek.Ivar jacobson,Grady Booch,James Rumbaugh menulis buku tentang UML

3.1.2 Definisi Objek

Sebuah objek merupakan sesuatu yang mempunyai keadaan, kelakuan dan identitas. Keadaan dari objek adalah satu dari kondisi yang memungkinkan dimana objek dapat muncul, dan dapat secara normal berubah berdasarkan waktu. Keadaan ini diimplementasikan dengan kelompok propertinya (disebut atribut), berisi nilai dari properti tersebut, ditambah keterhubungan objek yang mungkin dengan objek lainnya. Kelakuan menentukan bagaimana sebuah objek beraksi dan bereaksi terhadap permintaan dari objek lainnya. Direpresentasikan dengan kelompok pesan yang direspon oleh objek (operasi yang dilakukan oleh objek). Kelakuan dari objek mendeskripsikan segala sesuatu yang dapat kita lakukan terhadap objek tersebut dan segala sesuatu yang dapat dilakukan

oleh objek untuk kita. Setiap objek mempunyai identitas yang unik. Identitas yang unik ini membuat kita dapat membedakan dua objek yang berbeda, walaupun kedua objek tersebut mempunyai keadaan dan nilai yang sama pada atributnya.

3.1.3 Karakteristik dasar orientasi objek

Sistem berorientasi objek berfokus untuk menangkap struktur dan perilaku dari sistem informasi dalam modul kecil yang mencakup baik data dan proses. Modul-modul kecil ini dikenal sebagai objek. Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama : ***Enkapsulasi***, ***Inheritance*** dan ***Polmorfisme***

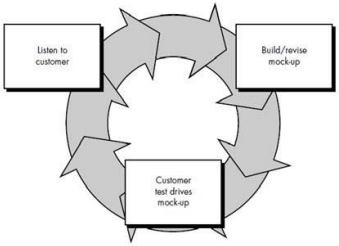
4 Pertemuan 4

4.1 Paradigma pengembangan system

pengembangan sistem informasi yang paling tua adalah siklus hidup sistem (*systems life cycle*), yang mengharuskan tahap-tahap formal dalam pengembangan sistem informasi. Tahap-tahap pengembangan sistem harus dilaksanakan secara urut dengan output-output yang telah ditentukan; setiap tahap mengharuskan persetujuan formal sebelum tahap selanjutnya dimulai. Metode siklus hidup sistem berguna untuk proyek-proyek besar yang memerlukan ketentuan dan spesifikasi formal dan pengendalian manajemen yang ketat pada setiap tahap pengembangan sistem informasi, tetapi metode siklus hidup sistem ini sangat bertele-tele dan mahal.

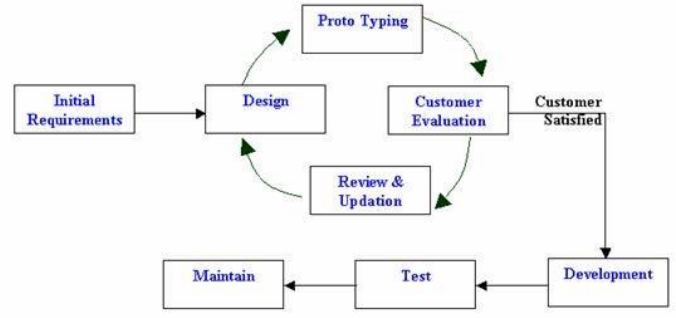
4.1.1 Paradigma prototype

Metode Prototype merupakan suatu paradigma baru dalam metode pengembangan perangkat lunak dimana metode ini tidak hanya sekedar evolusi dalam dunia pengembangan perangkat lunak, tetapi juga merevolusi metode pengembangan perangkat lunak yang lama yaitu sistem sekuensial yang biasa dikenal dengan nama SDLC atau waterfall development model.



Dalam Model Prototype, prototype dari perangkat lunak yang dihasilkan kemudian dipresentasikan kepada pelanggan, dan pelanggan tersebut diberikan kesempatan untuk memberikan masukan sehingga perangkat lunak yang dihasilkan nantinya betul-betul sesuai dengan keinginan dan kebutuhan pelanggan.
Perubahan dan presentasi prototype dapat dilakukan berkali-kali sampai dicapai kesepakatan bentuk dari perangkat lunak yang akan dikembangkan.
Teknik – teknik Prototyping Meliputi :

- Perancangan Model
- Perancangan Dialog
- Simulasi
  Berikut adalah 4 langkah yang menjadi karakteristik dalam proses pengembangan pada metode prototype, yaitu :
- Pemilihan fungsi
- Penyusunan Sistem Informasi
- Evaluasi
- Penggunaan Selanjutnya
  Metode ini menyajikan gambaran yang lengkap dari suatu sistem perangkat lunak, terdiri atas model kertas, model kerja dan program. Pihak pengembang akan melakukan identifikasi kebutuhan pemakai, menganalisa sistem dan melakukan studi kelayakan serta studi terhadap kebutuhan pemakai, meliputi model interface, teknik prosedural dan teknologi yang akan dimanfaatkan.



Proto Type Model

Berikut adalah Tahapan – tahapan Proses Pengembangan dalam Model Prototype, yaitu :

- **Pengumpulan kebutuhan**
  Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.
- **Membangun prototyping**
  Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output).
- **Evaluasi protoptyping**
  Evaluasi ini dilakukan oleh pelanggan, apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan pelanggan atau belum. Jika sudah sesuai, maka langkah selanjutnya akan diambil. Namun jika tidak, prototyping direvisi dengan mengulang langkah-langkah sebelumnya.
- **Mengkodekan sistem**
  Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.
- **Menguji sistem**
  Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, kemudian dilakukan proses Pengujian. Pengujian ini dilakukan dengan White Box, Black Box, Basis Path, pengujian arsitektur, dll.
- **Evaluasi Sistem**
  Pelanggan mengevaluasi apakah perangkat lunak yang sudah jadi sudah sesuai dengan yang diharapkan . Jika ya, maka proses akan dilanjutkan ke tahap selanjutnya, namun jika perangkat lunak yang sudah jadi tidak/belum sesuai dengan apa yang diharapkan, maka tahapan sebelumnya akan diulang.
- **Menggunakan sistem**
  Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.
  Model Prototyping ini sangat sesuai diterapkan untuk kondisi yang beresiko tinggi di mana masalah-masalah tidak terstruktur dengan baik, terdapat fluktuasi kebutuhan pemakai yang berubah dari waktu ke waktu atau yang tidak terduga, bila interaksi dengan pemakai menjadi syarat mutlak dan waktu yang tersedia sangat terbatas sehingga butuh penyelesaian yang segera. Model ini juga dapat

berjalan dengan maksimal pada situasi di mana sistem yang diharapkan adalah yang inovatif dan mutakhir sementara tahap penggunaan sistemnya relatif singkat.

Berikut merupakan Jenis – jenis dari Prototyping :

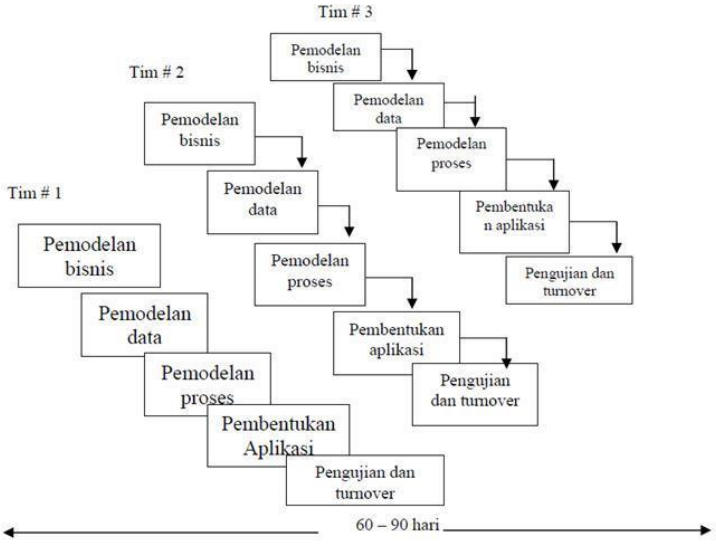
- **Feasibility prototyping**
  digunakan untuk menguji kelayakan dari teknologi yang akan digunakan untuk system informasi yang akan disusun.
- **Requirement prototyping**
  digunakan untuk mengetahui kebutuhan aktivitas bisnis user.
- **Desain Prototyping**
  digunakan untuk mendorong perancangan sistem informasi yang akan digunakan.
- **Implementation prototyping**
  merupakan lanjutan dari rancangan prototype, prototype ini langsung disusun sebagai suatu sistem informasi yang akan digunakan.
- **Contoh Penerapan Metode Prototype.**
  Sebuah rumah sakit ingin membuat aplikasi sistem database untuk pendataan pasiennya. Seorang atau sekelompok programmer akan melakukan identifikasi mengenai apa saja yang dibutuhkan oleh pelanggan, dan bagaimana model kerja program tersebut. Kemudian dilakukan rancangan program yang diujikan kepada pelanggan. Hasil/penilaian dari pelanggan dievaluasi, dan analisis kebutuhan pemakai kembali di lakukan.

- **Kelebihan Model Prototype :**
  - Pelanggan berpartisipasi aktif dalam pengembangan sistem, sehingga hasil produk pengembangan akan semakin mudah disesuaikan dengan keinginan dan kebutuhan pelanggan.
  - Penentuan kebutuhan lebih mudah diwujudkan.
  - Mempersingkat waktu pengembangan produk perangkat lunak.
  - Adanya komunikasi yang baik antara pengembang dan pelanggan.
  - Pengembang dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan.
  - Lebih menghemat waktu dalam pengembangan sistem.
  - Penerapan menjadi lebih mudah karena pelanggan mengetahui apa yang diharapkannya.

- **Kekurangan Model Prototype :**
  - Proses analisis dan perancangan terlalu singkat.
  - Biasanya kurang fleksibel dalam mengahadapi perubahan.
  - Walaupun pemakai melihat berbagai perbaikan dari setiap versi prototype, tetapi pemakai mungkin tidak menyadari bahwa versi tersebut dibuat tanpa memperhatikan kualitas dan pemeliharaan jangka panjang.
  - Pengembang kadang-kadang membuat kompromi implementasi dengan menggunakan sistem operasi yang tidak relevan dan algoritma yang tidak efisien.

4.1.2 Paradigma RAD (Rapid Application *Development*)

Rapid Aplication Development (RAD) adalah sebuah model proses perkembanganperangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek (kira-kira 60 sampai 90 hari). Model RAD ini merupakan sebuah adaptasi "kecepatan tinggi" dari model sekuensial linier dimana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen.



Berikut adalah Tahapan – tahapan Proses Pengembangan dalam Model Rapid Application Development (RAD), yaitu :

- **Bussiness Modeling**
  Fase ini untuk mencari aliran informasi yang dapat menjawab pertanyaan berikut:
- Informasi apa yang menegndalikan proses bisnis?
- Informasi apa yang dimunculkan?
- Di mana informasi digunakan ?
- Siapa yang memprosenya ?
- **Data Modeling**
  Aliran informasi yang didefinisikan sebagai bagian dari fase bussiness modeling disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik (atribut) masing-masing objek diidentifikasi dan hubungan antar objek-objek tersebut didefinisikan.
- **Proses Modeling**
  Aliran informasi yang didefinisikan di dalam fase data modeling ditransformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.
- **Aplication Generation**
  Selain menggunakan bahasa pemrograman generasi ketiga, RAD juga memakai komponen program yang telah ada atau menciptakan komponen yang bisa dipakai lagi. Ala-alat bantu bisa dipakai untuk memfasilitasi konstruksi perangkat lunak.
- **Testing dan Turnover**
  Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tetapi komponen baru harus diuji dan semua interface harus dilatih secara penuh.

- **Kelebihan Model RAD :**
  - Lebih efektif dari Pengembangan Model waterfall/sequential linear dalam menghasilkan sistem yang memenuhi kebutuhan langsung dari pelanggan.
  - Cocok untuk proyek yang memerlukan waktu yang singkat.

- Model RAD mengikuti tahap pengembangan sistem seperti pada umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada sehingga pengembang tidak perlu membuatnya dari awal lagi sehingga waktu pengembangan menjadi lebih singkat dan efisien.

- **Kekurangan Model RAD :**
  - Model RAD menuntut pengembangan dan pelanggan memiliki komitmen di dalam aktivitas rapid-fire yang diperlukan untuk melengkapi sebuah sistem, di dalam kerangka waktu yang sangat diperpendek. Jika komitmen tersebut tidak ada, proyek RAD akan gagal.
  - Tidak semua aplikasi sesuai untuk RAD, bila system tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat bermasalah.
  - RAD tidak cocok digunakan untuk sistem yang mempunyai resiko teknik yang tinggi.
  - Membutuhkan Tenaga kerja yang banyak untuk menyelesaikan sebuah proyek dalam skala besar.
  - Jika ada perubahan di tengah-tengah pengerjaan maka harus membuat kontrak baru antara pengembang dan pelanggan.

4.1.3 Paradigma extreme programming

metode rekayasa perangkat lunak pendekatan berorientasi objek model *Extreme Programming* (XP). Paradigma pembangunan mencakup seperangkat aturan dan praktik yang terjadi dalam konteks kerangka empat kegiatan yaitu: perencanaan, desain, coding, dan pengujian. Keempat aktivitas inilah yang akan menghasilkan sebuah perangkat lunak yang didasari dengan konsep model *Extreme Programming*. (Pressman 2010).

1. Perencanaan (*Planning*)

   Dalam tahap ini dikumpulkan kebutuhan awal *user* atau dalam XP disebut *user stories.* Hal ini dibutuhkan agar pengembang mengerti bisnis konten, kebutuhan *output* sistem, dan fitur utama dari *software* yang dikembangkan. Tahapan ini untuk menganalisa kebutuhan dari sistem tersebut untuk dapat digunakan sesuai dengan *user requirement* atau *user stories.*

2. Desain (*Design*)

   Desain dari sistem pada penelitian ini digambarkan dengan model UML berupa *use case diagram, activity diagram,* dan *relation table.* Pembuatan desain pada XP tetap mengedepankan prinsip *Keep it Simple (KIS).* Desain disini merupakan representasi dari sistem guna mempermudah pengembang dalam membangun sistem. *Desain* ini dimaksudkan untuk mempermudah pengembangan sistem nantinya.

3. *Coding*

   Proses melakukan *coding system* (pengkodean perangkat lunak) oleh *Programmer/Software Engineer* sesuai dengan *planning* dan *design* yang telah dibuat sebelumnya.

4. *Testing*

   Tahap ini akan menggunakan *unit test* yang sebelumnya telah dibuat. Karena pembuatan dari unit test adalah pendekatan utama dari XP. Dalam melakukan pengujian, penulis menggunakan 2 teknik pengujian yaitu pengujian *white box.* Pada tahap pengujian *black box,* dilakukan pengujian setiap unit test, maksudnya melakukan pengujian integrasi antara input dan hasil output yang sesuai semestinya terjadi.

4.1.4 Paradigma Inkremental

Model incremental menggabungkan elemen-elemen model sekuensial linier (diimplementasikan secara berulang) dengan filosofi prototype interatif. Model ini memakai urutan-urutan linier di dalam model yang membingungkan, seiring dengan laju waktu kalender. Setiap urutan linier menghasilkan pertambahan perangkat lunak yang kemudian dapat disampaikan kepada pengguna.

Pada saat model incremental (pertambahan) ini digunakan, pertambahan pertama sering merupakan produk inti (*core product),* yaitu sebuah model pertambahan yang dipergunakan, tetapi beberapa muka tambahan (beberapa diketahui dan beberapa tidak) tetap tidak disampaikan. Produk inti tersebut dipergunakan oleh pelanggan (atau mengalami pengkajian detail). Sebagai hasil dari pemakaian dan/atau evaluasi maka dikembangkan rencana bagi pertambahan selanjutnya. Rencana tersebut menekankan modifikasi produk inti untuk secara lebih

baik memenuhi kebutuhan para pelanggan dan penyampaian fitur serta fungsional tambahan. Proses ini mengikuti penyampaian setiap pertambahan sampai bisa menghasilkan produk yang lengkap.

Model proses incremental tersebut, seperti model prototype dan pendekatan-pendekatan evolusioner yang lain, bersifat iterative. Tetapi tidak seperti model prototype, model pertambahan berfokus pada penyampaian produk operasional dalam setiap pertambahannya. Pertambahan awal ada di versi *stripped down* dari produk akhir, tetapi memberikan kemampuan untuk melayani pemakai dan juga menyediakan platform untuk evaluasi oleh pemakai.

Perkembangan pertambahan, khususnya berguna pada saat *staffing*, tidak bisa dilakukan dengan menggunakan implementasi lengkap oleh batasan waktu bisnis yang sudah disepakati untuk proyek tersebut. Jika produk inti diterima dengan baik, maka staf tambahan (bila dibutuhkan) bisa ditambahkan untuk mengimplementasi pertambahan selanjutnya. Sebagai tambahan, system mayor yang sedang pada masa perkembangan serta waktu penyampaiannya belum pasti, mungkin membutuhkan keberadaan perangkat keras yang baru. Bisa juga rencana tertentu dibuat untuk menghindari pemakaian perangkat lunak ini, sehingga memungkinkan fungsionalitas partial disampaikan kepada pemakai tanpa harus banyak tertunda.

4.1.5 Penggunaan Agility

Kelincahan (*Agility*) dalam kinerja bisnis adalah kemampuan perusahaan untuk sejahtera dalam pasar global yang berubah cepat dan terus terfragmen unutk produk dan jasa berkualitas tinggi, berkinerja baik, dan disesuaikan dengan pelanggan. Perusahaan yang lincah dapat membuat laba dalam pasar dengan pilihan produk yang luas dan bermasa hidup pendek, dan dapat memproduksi pesanan secara individual dan dengan jumlah yang besar. Perusahaan tersebut mendukung penyesuaian massal (*mass customization*) dengan menawarkan produk individual sambil mempertahankan produksi dalam volume yang tinggi. Perusahaan yang lincah sangat bergantung pada tegnologi Internet untuk memadukan dan mengelola proses bisnis, sambil

menyediakan daya pemrosesan informasi untuk melayani banyak pelanggan sebagai individual.

Ada empat strategi dasar yang harus diimplementasikan untu menjadi perusahaan yang lincah. Antara lain :

1. Pelanggan dari perusahaan yang lincah menganggap produk atau jasa sebagai solusi terhadap masalah individual mereka. Jadi, harga produk dapat ditentukan berdasarkan biaya produksinya.
2. Perusahaan yang lincah bekerja sama dengan pelanggan, pemasok dan perusahaan lain bahkan dengan pesaing. Hal ini memungkinkan perusahaan untuk memasarkan prosuk dengan cepat dan hemat, dimanapun sumber daya berada dan siapapun yang memlikinya.
3. Perusahaan yang lincah dapat bertahan ketika terjadi perubahan dan ketidakpastian. Perusahaan menggunakan struktur organisasi yang fleksibel sehingga sesuai dengan peluang pelanggan yang terus berubah dan berbeda-beda.

Akhirnya perusahaan yang lincah dapat meningkatkan dampak sumberdaya manusia dan pengetahuan yang mereka miliki. Dengan memelihara semangat wira usaha, perusahaan yang lincah dapat memberikan insentif yang tinggi bagi tanggung jawab, kemampuan beradaptasi, dan inovasi pegawai.

Cara lain untuk memikirkan mengenai kelincahan dalam bisnis. Kerangka kerja ini menekankan pada peran yang dapat dimainkan oleh pelanggan, mitra bisnis dan tegnologi informasi dalam mengembangkan dan mempertahankan kelincahan strategis perusahaan. Perhatikan bagaimana tegnologi informasi dapat memungkinkan perusahaan untuk mengembangkan hubungan dengan pelanggan dalam komunitas virtual yang membantu perusahaan untuk bermitra dengan pemasok, distributor, manufaktur kontrak dan pihak lainnya melalui portal kerja sama dan sistem rantai pasokan berbasis Web lainnya yang secara signifikan memperbaiki kelincahan perusahaan dalam melihat peluang bisnis yang inovatif.

5 Pertemuan 5

5.1 Metodologi Analisis dan Perancangan berorientasi Objek

Metodologi ini mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

5.1.1 RUP (Rational Unified Process)

RUP, singkatan dari Rational Unified Process, adalah suatu kerangka kerja proses pengembangan perangkat lunak iteratif yang dibuat oleh Rational Software, suatu divisi dari IBM sejak 2003. RUP bukanlah suatu proses tunggal dengan aturan yang konkrit, melainkan suatu kerangka proses yang dapat diadaptasi dan dimaksudkan untuk disesuaikan oleh organisasi pengembang dan tim proyek perangkat lunak yang akan memilih elemen proses sesuai dengan kebutuhan mereka.

RUP menggunakan konsep object oriented, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan Unified Model Language(UML). Melalui gambar dibawah dapat dilihat bahwa RUP memiliki, yaitu:

a. Dimensi pertama di gambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu major milestoneyang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas Inception, Elaboration, Construction, dan Transition.

b. Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari

empat elemen penting, yakni who is doing, what, how dan when.

Dimensi ini terdiri atas:

Business Modeling, Requirement, Analysis and Design, Implementation, Test,

Deployment, Configuration dan Change Manegement, Project Management, Environtment.

Pada penggunaan kedua standard tersebut diatas yang berorientasi obyek (Object Oriented) memiliki menfaat yakni:

a. Improve productivity
   standard ini dapat memanfaatkan kembali komponen-komponen yang telah tersedia/dibuat sehingga dapat meningkatkan produktifitas.
b. Deliver hight quality system
   kualitas sistem informasi dapat ditingkatkan sebagai sistem yang telah dibuat pada komponen-komponen yang telah teruji (well -tested dan well -proven) sehingga dapat mempercepat delivery sistem informasi yang telah dibuat dengan kualitas yang tinggi.
c. Lower maintenance cost
   Standard ini dapat membantu untuk meyakinkan dampak perubahan yang teralokasi dan masalah dapat dengan mudah terdeteksi sehingga hasilnya biaya pemeliharaan dapat dioptimalkan atau lebih rendah dengan pengembangan informasi tanpa standar yang jelas.
d. Facilitate reuse
   Standard ini memiliki kamampuan yang mengembangkan komponen-komponen yang dapat digunakan kembali untuk pengembangan aplikasi yang lainnya.
e. Manage complexity
   Standard ini mudah untuk mengatur dan monitor semua proses dari semua tahapan yang ada sehingga suatu pengembangan sistem informasi yang amat kompleks dapat dilakukan dengan aman sesuai dengan harapan semua manager proyek IT/IS yakni deliver good quality software within cost and schedule time and the users accepted.

Fase RUP

1. Inception/insepsi
   - ✓ Menentukan Ruang lingkup proyek
   - ✓ Membuat 'Business Case'
   - ✓ Menjawab pertanyaan 'apakah yang dikerjakan dapat menciptakan 'good business sense' sehingga proyek dapat dilanjutkan

2.Elaboration/elaborasi
   - ✓ Menganalisa berbagai persyaratan dan resiko
   - ✓ Menetapkan ‘Base Line’
   - ✓ Merencanakan fase berikutnya yaitu construction

3. Construction/kontruksi
   - ✓ Melakukan sederetan iterasi
   - ✓ Pada setiap iterasi akan melibatkan prose berikut : analisa desain, implementasi dan testing

4. Transition/Transisi
   - ✓ Membuat apa yang sudah dimodelkan menjadi suatu produk jadi.
     Dalam fase ini dilakukan:
     - Beta dan performance testing
     - Membuat dokumentasi tambahan seperti: training, user guide dan sales kit
     - Membuat rencana peluncuran produk ke komunitas pengguna

Peran Use Case Pada Setiap Fase

1. inception
   - Menolong mengembangkan scope proyek
   - Menolong menetapkan penjadwalan dan anggaran
2. Elaboration
   - Menolong dalam melakukan analisa resiko
   - Menolong mempersiapkan fase berikutnya yaitu konstruksi
3. Construction
   - Melakukan sederetan iritasi
   - Pada setiap iterasi akan melibatkan proses berikut: analisa desain, implementasi dan testing

4. Transition
    - Membuat apa yang sudah dimodelkan menjadi suatu produk jadi
    - Dalam fasi ini dilakukan:
        a. Beta dan performance testing
        b. Membuat dokumentasi tambahan seperti: training, user guide dan sales kit
        c. Membuat rencana peluncuran produk ke komunitas pengguna

Penerapan Tahapan Metodologi Pengembangan Lunak dengan Menggunakan RUP (Contoh Kasus) Metodologi *Rational Unified Process (RUP).*Metode RUP merupakan metode pengembangan kegiatan yang berorientasi pada proses. Dalam metode ini, terdapat empat tahap pengembangan perangkat lunak yaitu :

1. Inception
   Pada tahap ini pengembang mendefinisikan batasan kegiatan, melakukan analisis kebutuhan user, dan melakukan perancangan awal perangkat lunak (perancangan arsitektural dan user case). Pada akhir fase ini, prototipe perangakat lunak versi Alpha harus sudah dirilis.
2. Elaboration
   Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikan fitur perangkat lunak hingga perilisan prototipe versi Betha dari perangkat lunak.
3. Contruction
   Pengimplentasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta dokumentasi perangkata lunak.
4. Transition
   Instalasi, deployment dan sosialisasi perangkat lunak dilakukan pada tahap ini.

6 Pertemuan 6

6.1 Konsep Dasar UML

Menurut Nugroho (2010:10), Sesungguhnya tidak ada batasan yag tegas diantara berbagai konsep dan konstruksi dalam UML, tetapi untuk menyederhanakannya, kita membagi sejumlah besar konsep dan dalam UML menjadi beberapa *view*. Suatu *view* sendiri pada dasarnya merupakan sejumlah konstruksi pemodelan UML yang merepresentasikan suatu aspek tertentu dari sistem atau perangkat lunak yang sedang kita kembangkan. Pada peringkat paling atas, *view-view* sesungguhnya dapat dibagi menjadi tiga area utama, yaitu: klasifikasi struktural (*structural classification*), perilaku dinamis (*dinamic behaviour*), serta pengolahan atau manajemen model (*model management*).

6.1.1 Definisi UML

Menurut Widodo, (2011:6), "UML adalah bahasa pemodelan standar yang memiliki sintak dan semantik".

Menurut Nugroho (2010:6), "UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek)." Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Berdasarkan pendapat yang dikemukakan di atas dapat ditarik kesimpulan bahwa UML adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis Objek (*Object Oriented programming*).

6.1.2 Diagram

Dalam UML sendiri terdapat beberapa diagram yang wajib dikuasai yaitu:

a. Structural Diagram

- Class Diagram, diagram ini terdiri dari *class*, *interface*, *association*, dan *collaboration*. Diagram ini menggambarkan objek - objek yang ada di sistem.
- Object Diagram, diagram ini menggambarkan hasil instansi dari *class diagram*. Diagram ini digunakan untuk membuat *prototype*.

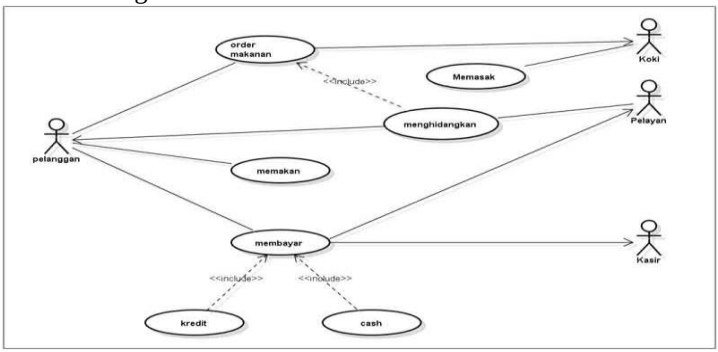
- Component Diagram, diagram ini menggambarkan kumpulan komponen dan hubungan antar komponen. Komponen terdiri dari *class*, *interface*, atau *collaboration*
- Deployment Diagram, diagram ini menggambarkan kumpulan *node* dan hubungan antar *node*. *Node* adalah entitas fisik dimana komponen di-*deploy*. Entitas fisik ini dapat berupa *server* atau perangkat keras lainnya.

b. Behavioral Diagram

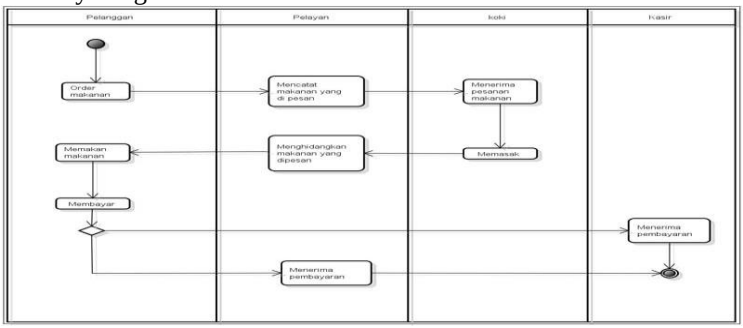
- Use case Diagram, diagram ini menggambarkan kumpulan *use case*, aktor, dan hubungan mereka. *Use case* adalah hubungan antara fungsionalitas sistem dengan aktor internal/eksternal dari sistem.
- Sequence Diagram, diagram ini menggambarkan interaksi yang menjelaskan bagaimana pesan mengalir dari objek ke objek lainnya.
- Collaboration Diagram, diagram ini merupakan bentuk lain dari *sequence diagram*. Diagram ini menggambarkan struktur organisasi dari sistem dengan pesan yang diterima dan dikirim.
- Statechart Diagram, diagram ini menggambarkan bagaimana sistem dapat bereaksi terhadap suatu kejadian dari dalam atau luar. Kejadian (*event*) ini bertanggung jawab terhadap perubahan keadaan sistem.
- Activity Diagram, menggambarkan aliran kontrol sistem. Diagram ini digunakan untuk melihat bagaimana sistem bekerja ketika dieksekusi.

6.1.3 View

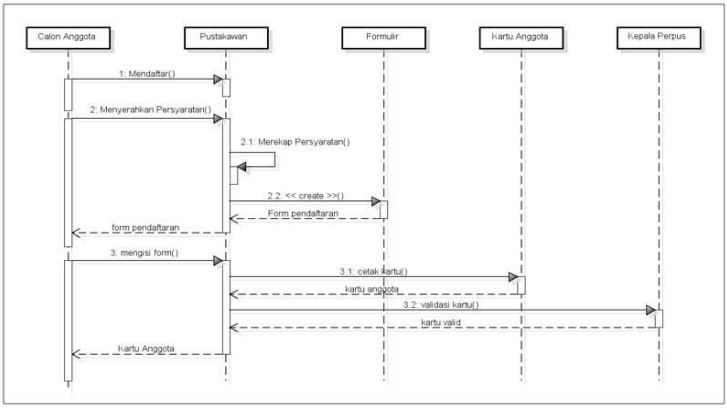
Use case diagram



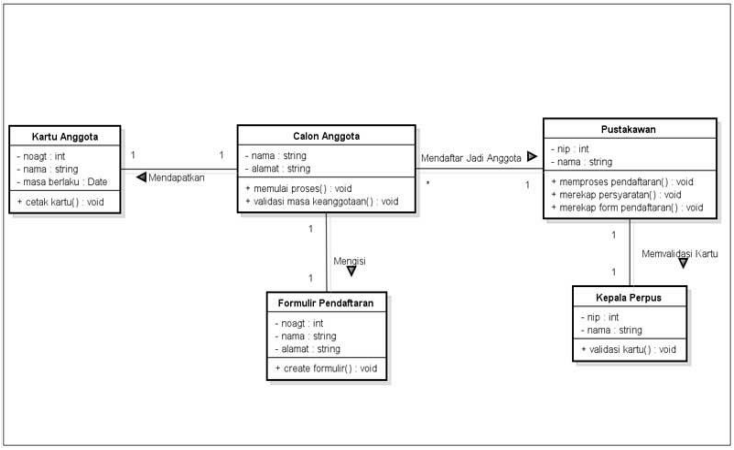
Activity Diagram



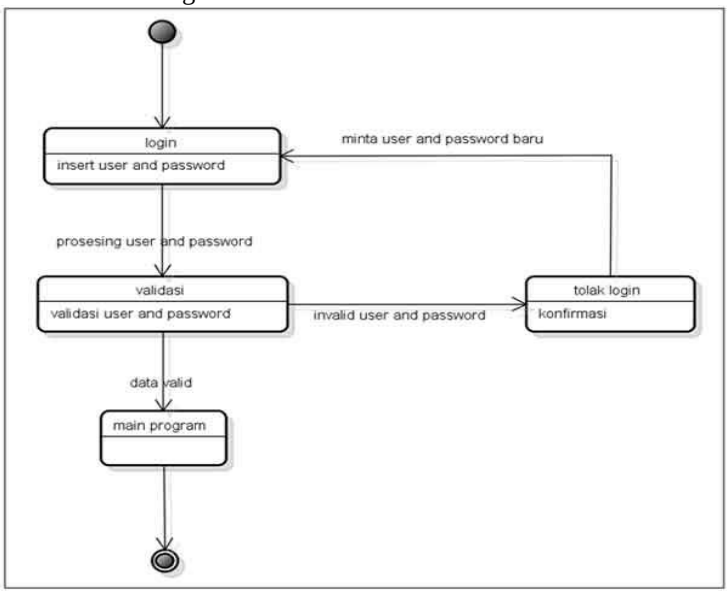
Sequence diagram



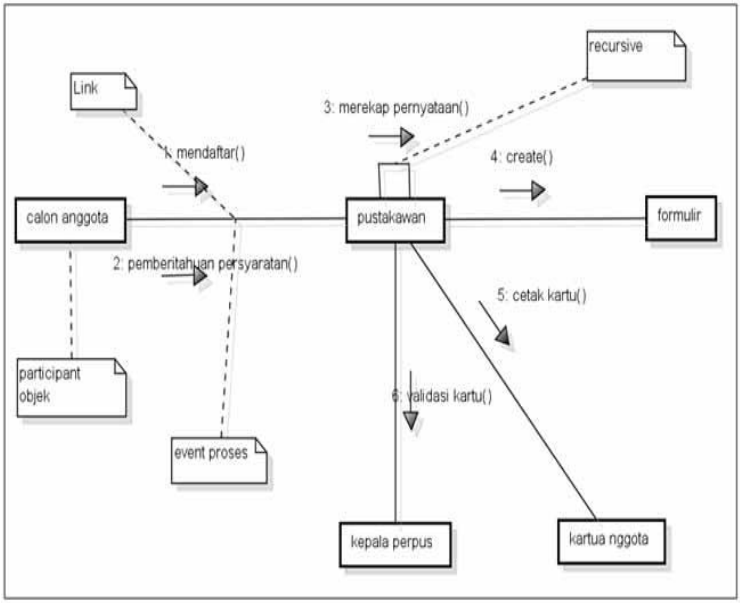
Class diagram



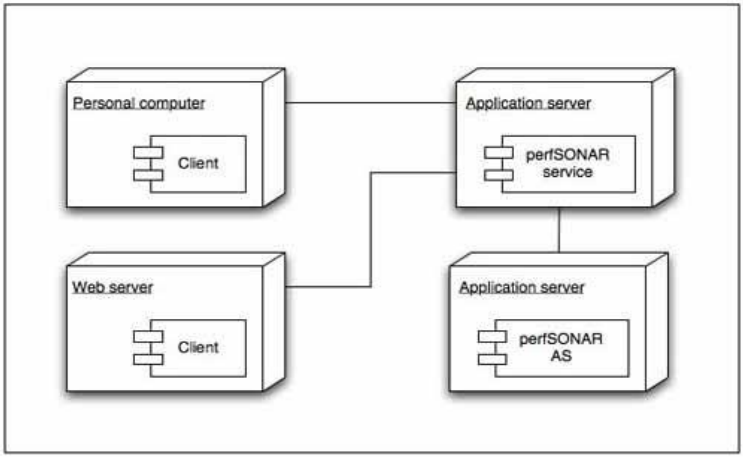
Statemachine diagram



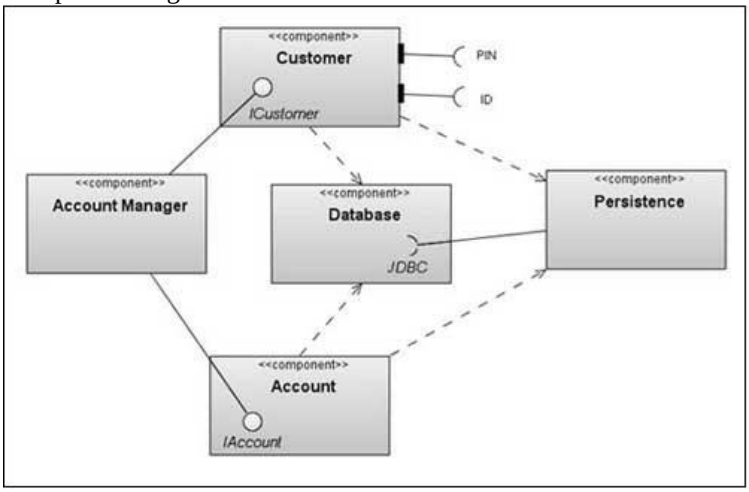
Communication diagram



Deployment diagram



Component diagram



7 Pertemuan 7

7.1 Design Pattern

Pattern adalah solusi umum yang dapat digunakan kembali pada permasalahan umum yang sering terjadi pada software design. Design pattern bukan desain final yang dapat ditransformasikan secara langsung kedalam kode. Ini hanyalah deskripsi atau template untuk mengetahui bagaimana menyelesaikan permasalahan yang dapat digunakan pada berbagai macam situasi yang berbeda. Design pattern dari object-oriented secara tipikal menunjukkan hubungan dan interaksi antara kelas dan objek tanpa menspesifikasikan kelas atau objek dari aplikasi final yang terlibat didalamnya.

7.1.1 Elemen Design Pattern

design pattern terdiri dari tiga elemen dasar, yaitu:

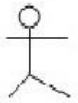
a. **Konteks**: situasi dimana *pattern* diterapkan (dan biasanya hal ini berulang)
b. **Masalah**: tujuan atau batasan yang akan dicapai oleh konteks
c. **Solusi**: desain umum dari masalah yang akan diselesaikan dan ditentukan batasannya

7.1.2 Diagram Use Case dan notasi

Diagram use case terdiri dari beberapa komponen, yaitu:

– Actor

Actor adalah perwakilan dari orang luar, proses atau hal yang berinteraksi
dengan sistem, subsistem ataupun class. Tiap actor berparti sipasi dengan satu atau lebih use-case. Actor berpartisipasi dengan use-case dengan pertukaran pesan. Actor dapat digambarkan seperti Gbr 1.



Gbr1. Notasi Actor

– Use Case

Use-case merupakan lingkupan sistem yang mengidentifikasikan hal-hal yang seharusnya dilakukan oleh sistem. Use-case berguna untuk menggambarkan suatu kelakuan dari sistem tanpa mengungkapkan struktur internal dari

sistem tersebut. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dll.

Gbr2. Notasi Use Case

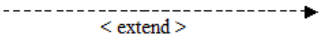
– Realationship

Relatioship berguna untuk menggambarkan hubungan antar aktor dan use- case dalam sistem. Di dalam diagram use case ada beberapa bentuk relationship, yaitu:

* *Association*, berfungsi sebagai jalur komunikasi antar actor dengan use case yang saling berpartisipasi. Association dapat dinotasikan dengan gambar garis lurus, seperti berikut ini:

Gbr3. Notasi Association

* *External*, berfungsi untuk menambahkan kelakuan tambahan ke dalam use case dasar yang tidak tahu menahu tentang hal tersebut. Relationship dapat digambarkan dengan notasi berikut:



Gbr4. Notasi External

* *Use case Generalization*, menggambarkan hubungan antara use case umum dengan use case yang lebih spesifik yang mewarisi dan menambah fitur terhadapnya. Relationship jenis akan digambarkan dalam bentuk notasi seperti berikut ini:

Gbr5. Notasi Generalization

* *Include*. Ini merupakan penambahan kelakuan tambahan ke dalam use case dasar yang secara eksplisit menjelaskan penambahannya.

< extend >

Gbr6. Notasi extend

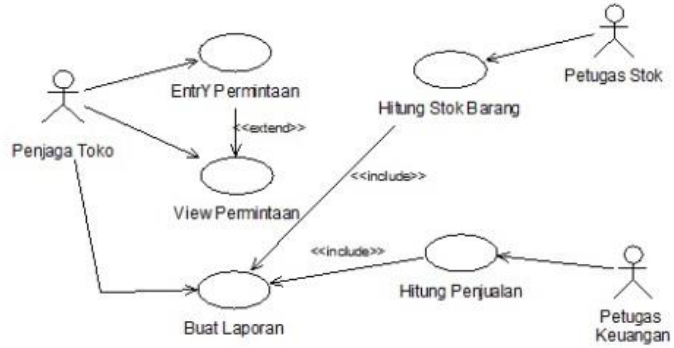
Diagram *Use Case* berguna dalam tiga hal :

• Menjelaskan fasilitas yang ada (*requirements*). Use Case baru selalu menghasilkan fasilitas baru ketika sistem di analisa, dan design menjadi lebih jelas.

• Komunikasi dengan klien. Penggunaan notasi dan simbol dalam diagram *Use Case* membuat pengembang lebih mudah berkomunikasi dengan klienkliennya.

• Membuat test dari kasus-kasus secara umum. Kumpulan dari kejadian-kejadian untuk *Use Case* bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

Contoh Diagram Use Case:



Gbr7. Diagram Use Case pada penjualan DVD

Pada contoh diagram use case di atas, ada terdapat 3 aktor, yaitu: penjaga toko, petugas stok, dan petugas keuangan. Sedangkan, use case-nya ada terdapat 5 buah yaitu: entry permintaan, hitung stok barang, buat laporan, view permintaan, dan hitung penjualan.

Penjaga toko akan melihat dan mencatat berapa banyak permintaan VCD dan membuat laporannya. Petugas Stok akan menghitung jumlah stok barang VCD) yang masih ada dan membuat laporannya. Sedangkan petugas keuangan akan menghitung berapa hasil penjualan dari VCD dan membuat laporan hasil penjualan tersebut.

7.1.3 Skenario

Setiap use case diagram dilengkapi dengan skenario, skenario use case / use case skenario adalah alur jalannya proses use case dari sisi aktor dan system. Berikut adalah format tabel skenario use case.

Nama Aktor	Reaksi Sistem
Skenario Normal	

Skenario Alternatif	

Skenario use case dibuat per use case terkecil, misalkan untuk generalisasi maka scenario yang dibuat adalah use case yang lebih khusus. Skenario normal adalah scenario bila system berjalan normal tanpa terjadi kesalahan atau error. Sedangkan skenario alternatif adalah scenario bila system tidak berjalan normal atau mengalami error. Skenario normal dan skenario alternatif dapat berjumlah lebih dari satu. Alur skenario inilah yang nantinya menjadi landasan pembuatan sequence diagram / diagram sekuen.

8 **Pertemuan 9**

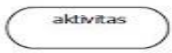
8.1 Diagram Aktivitas

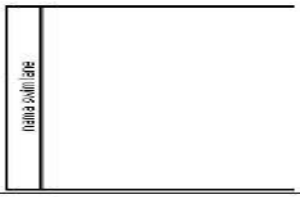
Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.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

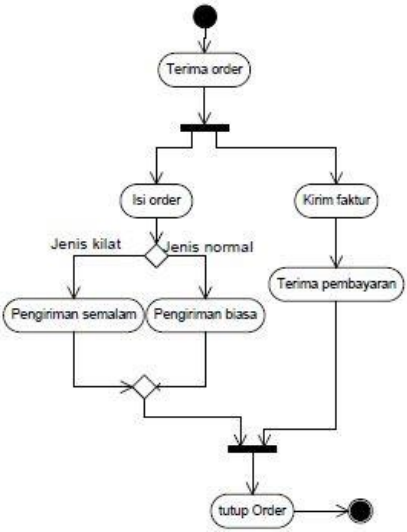
8.1.1 Diagram Aktivitas

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 yangdilakukan 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.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
   Gambar di bawah ini menunjukkan sebuah contoh sederhana dari diagram aktivitas untuk sistem Order. Diagram diawali dengan node status awal dankemudian 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.



9 Pertemuan 10

9.1 Diagram Interaksi

Dari nama *Interaksi* jelas bahwa diagram digunakan untuk menggambarkan beberapa jenis interaksi antara unsur-unsur yang berbeda dalam model. Jadi interaksi ini adalah bagian dari perilaku dinamis dari sistem.

Perilaku interaktif ini diwakili dalam UML oleh dua diagram yang dikenal sebagai *Sequence diagram* dan *diagram Kolaborasi.* Tujuan dasar dari kedua diagram serupa.

Sequence diagram menekankan pada urutan waktu pesan dan diagram kolaborasi menekankan pada organisasi struktural dari objek yang mengirim dan menerima pesan.

Tujuan dari diagram interaksi adalah untuk menggambarkan perilaku interaktif dari sistem. Sekarang memvisualisasikan interaksi adalah tugas yang sulit. Jadi solusinya adalah dengan menggunakan berbagai jenis model untuk menangkap aspek yang berbeda dari interaksi.

Itulah sebabnya urut dan kolaborasi diagram digunakan untuk menangkap sifat dinamis tapi dari sudut yang berbeda.

Jadi tujuan dari diagram interaksi bisa menggambarkan sebagai:

- Untuk menangkap perilaku dinamis dari suatu sistem.

- Untuk menggambarkan arus pesan dalam sistem.
- Untuk menggambarkan organisasi struktural dari objek.
- Untuk menggambarkan interaksi antara objek-objek.

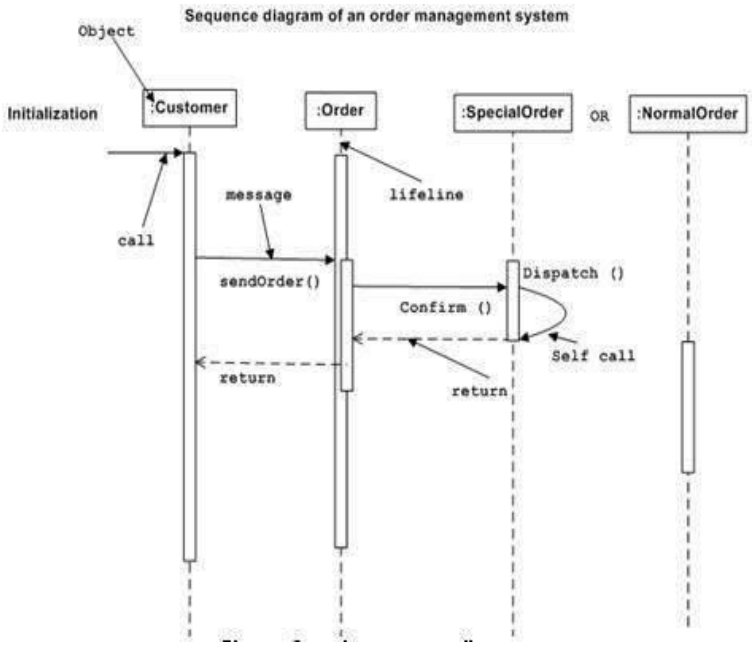
Seperti yang kita telah membahas bahwa tujuan dari diagram interaksi adalah untuk menangkap aspek dinamis dari sistem. Jadi untuk menangkap aspek dinamis kita perlu memahami apa aspek dinamis dan bagaimana divisualisasikan. Aspek dinamis dapat didefinisikan sebagai snap shot dari sistem yang sedang berjalan pada saat tertentu.
Kami memiliki dua jenis diagram interaksi dalam UML. Salah satunya adalah urutan diagram dan yang lainnya adalah diagram kolaborasi. Diagram urutan menangkap urutan waktu aliran pesan dari satu objek ke yang lain dan diagram kolaborasi menggambarkan organisasi dari objek dalam sistem mengambil bagian dalam aliran pesan.
Jadi hal-hal berikut ini untuk diidentifikasi secara jelas sebelum menggambar diagram interaksi:

- Benda mengambil bagian dalam interaksi.
- Pesan mengalir di antara benda-benda.
- urutan di mana pesan yang mengalir.
- organisasi objek.

Berikut adalah dua diagram interaksi sistem manajemen pesanan modeling. Diagram pertama adalah diagram urutan dan yang kedua adalah diagram kolaborasi.

9.1.1 Diagram Sequential

Diagram urutan adalah memiliki empat objek (Customer, Order, SpecialOrder dan NormalOrder).
Diagram berikut menunjukkan urutan pesan untuk *SpecialOrder* objek dan sama dapat digunakan dalam kasus *NormalOrder* objek. Sekarang penting untuk memahami urutan waktu pesan mengalir. Aliran pesan apa-apa tapi panggilan metode objek.
Panggilan pertama adalah *sendOrder ()* yang merupakan metode objek *Order*. Panggilan berikutnya adalah *konfirmasi ()* yang merupakan metode *SpecialOrder* objek dan panggilan terakhir adalah *Dispatch ()* yang merupakan metode *SpecialOrder* objek. Jadi di sini diagram terutama menggambarkan metode panggilan dari satu objek ke yang lain dan ini juga merupakan skenario yang sebenarnya ketika sistem berjalan.

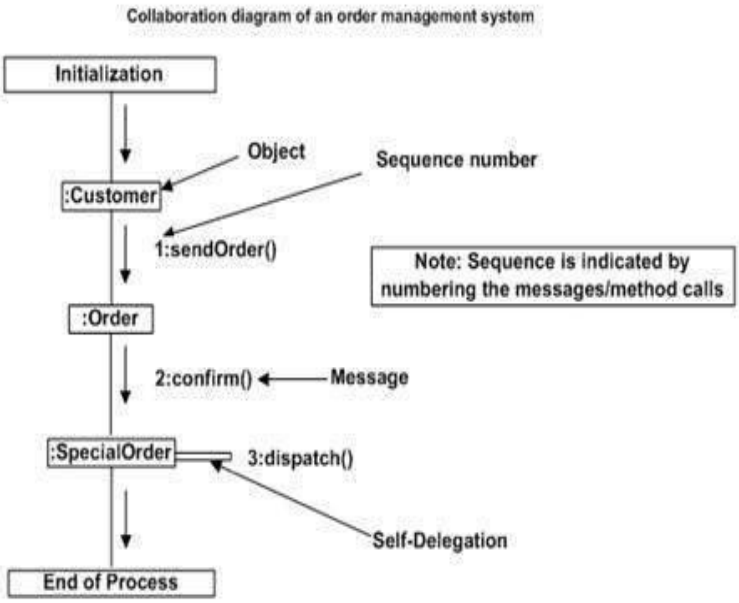


9.1.2 Diagram Colaboration

Diagram interaksi kedua adalah diagram kolaborasi. Ini menunjukkan organisasi objek seperti gambar di bawah. Di sini, di diagram kolaborasi metode urutan panggilan diindikasikan oleh beberapa teknik penomoran seperti yang ditunjukkan di bawah ini. Jumlah tersebut menunjukkan bagaimana metode yang disebut satu demi satu. Kami telah mengambil sistem manajemen pesanan yang sama untuk menggambarkan diagram kolaborasi.

Metode panggilan mirip dengan diagram urutan. Tetapi perbedaannya adalah bahwa diagram urutan tidak menggambarkan organisasi objek di mana sebagai diagram kolaborasi menunjukkan organisasi objek.

Sekarang untuk memilih antara dua diagram ini penekanan utama diberikan pada jenis kebutuhan. Jika waktu urutan penting maka urutan diagram digunakan dan jika organisasi diperlukan maka diagram kolaborasi digunakan.



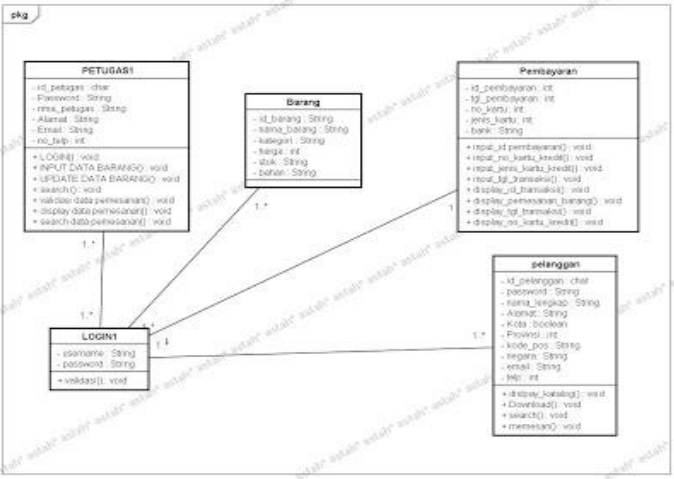
10 Pertemuan 11

10.1 Diagram Class dan Diagram Objek

10.1.1 Diagram Class

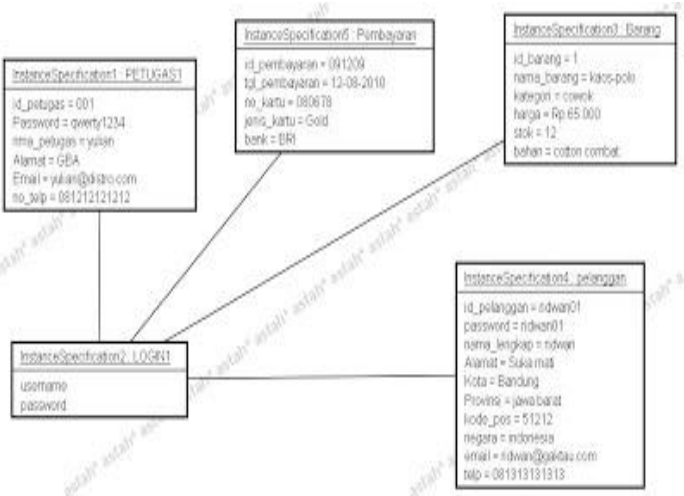
Class diagram adalah sebuah class yang menggambarkan struktur dan penjelasan class, paket, dan objek serta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Class diagram juga menjelaskan hubungan antar *class* dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan. Class juga memiliki 3 area pokok (utama) yaitu : nama,atribut,dan operasi. Nama berfungsi untuk member identitas pada sebuah kelas, atribut fungsinya adalah untuk member karakteristik pada data yang dimiliki suatu objek di dalam kelas, sedangkan operasi fungsinya adalah memberikan sebuah fungsi ke sebuah objek . Dalam mendefinisikan metode yang ada di dalam kelas harus diperhatikan yang namanya *Cohesion* dan *Coupling*, *Cohesion* adalah ukuran keterkaitan sebuah instruksi di sebuah metode, *Coupling* adalah ukuran keterkaitan antar metode. Di dalam class diagram terdapat hubungan antar kelas secara konseptual, yang disebut *Relasi*

antar Class, di UML disediakan macam-macam relasi antar Class, diantaranya: Asosiasi (Hubungan statis antar kelas), Agregasi (hubungan dari keseluruhan objek), Generalisasi (relasi beberapa subkelas ke super kelas), Dependency (keterhubungan tiap kelas.)



10.1.2 Diagram Objek

Object Diagram adalah Salah satu perancangan sistem yang digunakan untuk menjelaskan tentang nama obyek, atribut dan metode yang dipakai. Sebuah Object Diagram adalah gambaran dari objek-objek dalam sebuah system pada satu waktu. Diagram ini sering juga disebut sebagai Diagram Perintah, karena pada diagram ini perintah-perintah nya lebih ditonjolkan daripada kelasnya.



11 **Pertemuan 12**

11.1 Diagram Component

diagram komponen merupakan diagram yang digunakan untuk model aspek fisik dari suatu sistem.

aspek fisik adalah unsur-unsur seperti executable, perpustakaan, file, dokumen dll yang berada di node diagram sehingga komponen yang digunakan untuk memvisualisasikan organisasi dan hubungan antar komponen dalam suatu sistem. diagram ini juga digunakan untuk membuat sistem executable.

11.1.1 Struktur Program

diagram komponen yang sangat penting dari perspektif implementasi. Jadi tim implementasi aplikasi harus memiliki pengetahuan yang tepat dari rincian komponen.

Sekarang penggunaan diagram komponen dapat digambarkan sebagai:

- Model komponen dari suatu sistem.
- Model skema database.
- executables model aplikasi.
- kode sumber sistem model.

11.1.2 Diagram Component

diagram komponen adalah jenis khusus dari diagram di UML. Tujuannya adalah juga berbeda dari semua diagram lain yang

dibahas sejauh ini. Tidak menggambarkan fungsi sistem tetapi itu menggambarkan komponen yang digunakan untuk membuat fungsi mereka.

Jadi dari sudut diagram komponen yang digunakan untuk memvisualisasikan komponen fisik dalam suatu sistem. komponen ini perpustakaan, paket, file dll

diagram komponen juga dapat digambarkan sebagai implementasi pandangan statis dari sebuah sistem. pelaksanaan statis merupakan organisasi dari komponen-komponen pada saat tertentu.

Diagram komponen tunggal tidak dapat mewakili seluruh sistem tetapi koleksi diagram digunakan untuk mewakili keseluruhan.

Jadi tujuan dari diagram komponen dapat diringkas sebagai:

- Memvisualisasikan komponen dari suatu sistem.
- Membangun executable dengan menggunakan maju dan reverse engineering.
- Jelaskan organisasi dan hubungan dari komponen.

diagram komponen yang digunakan untuk menggambarkan artefak fisik dari suatu sistem. artefak ini termasuk file, executable, perpustakaan dll

Jadi tujuan diagram ini berbeda, diagram Komponen yang digunakan selama tahap implementasi aplikasi. Tapi itu dipersiapkan baik di muka untuk memvisualisasikan rincian pelaksanaan.

Awalnya sistem ini dirancang menggunakan diagram UML yang berbeda dan kemudian ketika artefak siap diagram komponen yang digunakan untuk mendapatkan ide dari pelaksanaan.

diagram ini sangat penting karena tanpa itu aplikasi tidak dapat diimplementasikan secara efisien. Sebuah diagram komponen siap juga penting untuk aspek-aspek lain seperti kinerja aplikasi, pemeliharaan dll.

Jadi sebelum menggambar diagram komponen artefak berikut harus diidentifikasi dengan jelas:

- File yang digunakan dalam sistem.
- Perpustakaan dan artefak lain yang relevan dengan aplikasi.
- Hubungan antara artefak.

Sekarang setelah mengidentifikasi artefak poin berikut perlu diikuti:

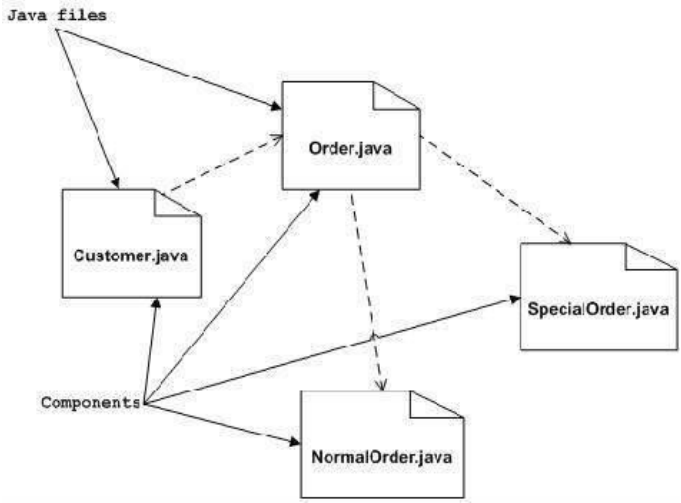
- Gunakan nama yang berarti untuk mengidentifikasi komponen yang diagram tersebut akan ditarik.
- Siapkan tata letak mental sebelum memproduksi menggunakan alat.
- Gunakan catatan untuk mengklarifikasi poin penting.

Berikut ini adalah diagram komponen untuk sistem manajemen pesanan. Berikut artefak adalah file. Jadi diagram menampilkan file dalam aplikasi dan hubungan mereka. Dalam aktual diagram komponen juga mengandung dll, perpustakaan, folder dll

Dalam diagram berikut empat file diidentifikasi dan hubungan mereka diproduksi. diagram komponen tidak dapat dicocokkan langsung dengan diagram UML lainnya dibahas sejauh ini. Karena ditarik untuk tujuan yang sama sekali berbeda.

Jadi diagram komponen berikut telah diambil mengingat semua poin yang disebutkan di atas:

Component diagram of an order management system



12 Pertemuan 13

12.1 Diagram Deployment

deployment diagram digunakan untuk memvisualisasikan topologi dari komponen fisik dari sebuah sistem dimana komponen software dikerahkan.

diagram sehingga penyebaran yang digunakan untuk menggambarkan statis lihat penyebaran sistem. deployment diagram terdiri dari node dan hubungan mereka.

12.1.1 Arsitektur Jaringan computer

deployment diagram terutama digunakan oleh para insinyur sistem. diagram ini digunakan untuk menggambarkan komponen fisik (hardware), distribusi dan hubungan mereka.
Untuk memperjelas secara rinci kita dapat memvisualisasikan diagram deployment sebagai komponen hardware / node yang komponen software berada.

aplikasi perangkat lunak yang dikembangkan untuk model proses bisnis yang kompleks. Hanya aplikasi perangkat lunak yang efisien tidak cukup untuk memenuhi kebutuhan bisnis. kebutuhan bisnis dapat digambarkan sebagai untuk mendukung peningkatan jumlah pengguna, waktu respon yang cepat dll.

Untuk memenuhi jenis kebutuhan komponen hardware harus dirancang secara efisien dan dengan cara yang efektif biaya.

Sekarang aplikasi perangkat lunak hari ini sangat kompleks di alam. aplikasi perangkat lunak dapat berdiri sendiri, berbasis web, didistribusikan, mainframe berbasis dan banyak lagi. Jadi, sangat penting untuk merancang komponen hardware efisien.

Jadi penggunaan diagram deployment dapat digambarkan sebagai berikut:

- Untuk model topologi hardware dari sistem.
- Untuk model sistem embedded.
- Untuk model rincian perangkat keras untuk sistem client / server.
- Untuk model detail hardware dari aplikasi terdistribusi.
- Maju dan reverse engineering.

12.1.2 Diagram Deployment

Nama *Deployment* sendiri menjelaskan tujuan diagram. deployment diagram digunakan untuk menggambarkan komponen perangkat keras di mana komponen software dikerahkan. diagram komponen dan diagram deployment yang erat terkait.

diagram komponen yang digunakan untuk menggambarkan komponen dan deployment diagram menunjukkan bagaimana mereka ditempatkan di hardware.

UML terutama dirancang untuk fokus pada artefak perangkat lunak dari suatu sistem. Tapi dua diagram ini adalah diagram khusus digunakan untuk fokus pada komponen perangkat lunak dan komponen perangkat keras.

Sehingga sebagian besar diagram UML digunakan untuk menangani komponen logis tapi deployment diagram yang dibuat untuk fokus pada topologi hardware dari sistem. deployment diagram yang digunakan oleh para insinyur sistem.

Tujuan dari diagram deployment dapat digambarkan sebagai:

- Memvisualisasikan topologi hardware dari sistem.
- Jelaskan komponen perangkat keras yang digunakan untuk menyebarkan komponen software.
- Jelaskan node pengolahan runtime.

diagram Deployment merupakan tampilan penyebaran sistem. Hal ini terkait dengan diagram komponen. Karena komponen dikerahkan menggunakan diagram deployment. Diagram deployment terdiri dari node. Node hanyalah hardware fisik yang digunakan untuk menyebarkan aplikasi.

deployment diagram berguna untuk insinyur sistem. Diagram deployment efisien sangat penting karena mengontrol parameter berikut

- prestasi
- skalabilitas
- rawatan
- portabilitas

Jadi sebelum menggambar diagram deployment artefak berikut harus diidentifikasi:

- node
- Hubungan antara node

Berikut diagram deployment adalah contoh untuk memberikan gambaran tentang pandangan penyebaran sistem manajemen pesanan. Di sini kita telah menunjukkan node sebagai:

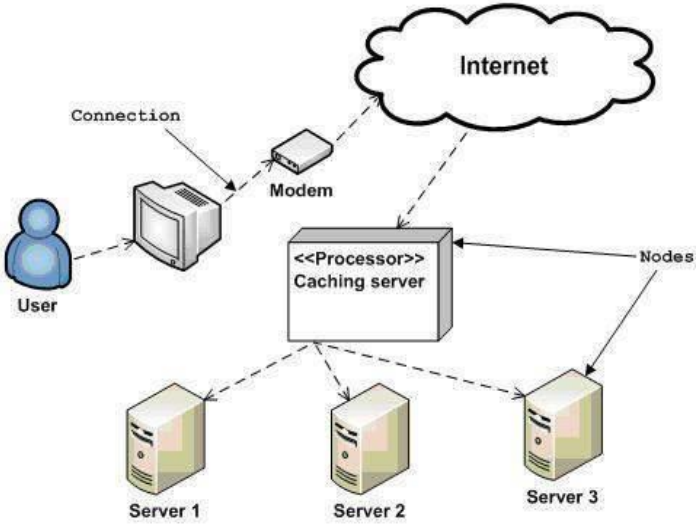
- Monitor

- Modem
- caching server
- Server

  Aplikasi ini diasumsikan aplikasi berbasis web yang digunakan dalam lingkungan cluster menggunakan server 1, Server 2 dan server 3. Pengguna menghubungkan ke aplikasi menggunakan internet. Kontrol yang mengalir dari server caching terhadap lingkungan berkerumun.

  Jadi diagram deployment berikut telah diambil mengingat semua poin yang disebutkan di atas:

Deployment diagram of an order management system



13 Pertemuan 14

13.1 Diagram State Machine

Menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu obyek pada sistem sebagai akibat dari stimulans yang diterima.

State Machine Diagram Untuk memodelkan behavior/methode (lifecycle) sebuah kelas atau object Memperlihatkan urutan kejadian sesaat (state) yang dilalui sebuah object, transisi dari sebuah state ke state lainnya.

13.1.1 Pengertian State Machine

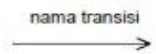
Diagram status atau state diagram atau statechart diagram menunjukkan kondisi yang dapat dialami atau terjadi pada sebuah objek sehingga setiap objek memiliki sebuah diagram status. Diagram status diadopsi dari penggambaran kondisi mesin status (state machine) yang menggambarkan status apa saja yang 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.

13.1.2 Notasi Diagram State Machine

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
	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

13.1.3 Pembuatan State Machine

Statechart diagram digunakan untuk menggambarkan negara objek yang berbeda dalam siklus hidupnya. Jadi penekanan diberikan pada perubahan negara atas beberapa peristiwa internal atau eksternal. Negara-negara ini benda penting untuk menganalisa dan menerapkan mereka secara akurat.

diagram statechart sangat penting untuk menggambarkan negara. Negara dapat diidentifikasi sebagai kondisi objek ketika peristiwa tertentu terjadi.

Sebelum menggambar diagram Statechart kita harus mengklarifikasi hal berikut:

- Mengidentifikasi objek penting untuk dianalisis.
- Mengidentifikasi negara.
- Mengidentifikasi peristiwa.

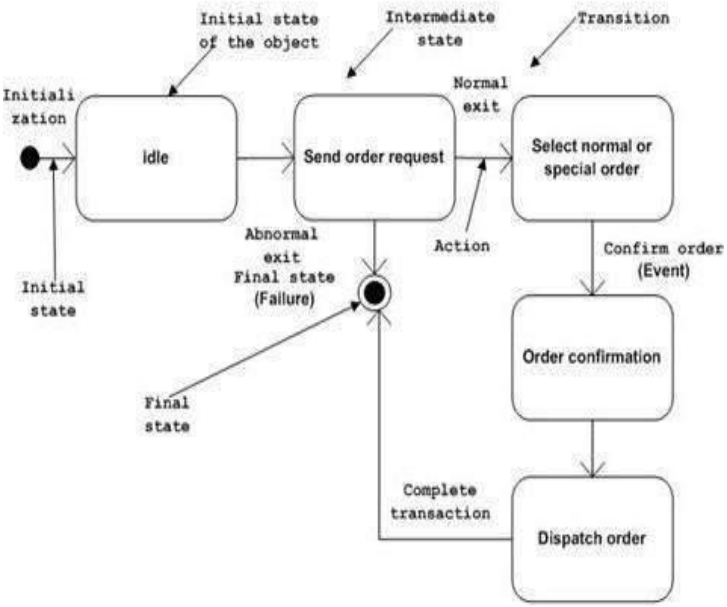
Berikut ini adalah contoh dari diagram Statechart mana keadaan objek *Orde* dianalisis.

Negara pertama adalah keadaan idle dari mana proses dimulai. Negara-negara berikutnya tiba untuk acara seperti *mengirim permintaan, mengkonfirmasi permintaan,* dan *ketertiban pengiriman.* Peristiwa ini bertanggung jawab untuk perubahan keadaan agar objek.

Selama siklus hidup dari sebuah objek (di sini memesan objek) ia pergi melalui negara-negara berikut dan mungkin ada beberapa yang tidak normal ada juga. keluar yang abnormal ini dapat terjadi karena beberapa masalah dalam sistem. Ketika seluruh siklus hidup lengkap dianggap sebagai transaksi yang lengkap seperti yang disebutkan di bawah ini.

Keadaan awal dan akhir dari sebuah objek juga ditunjukkan di bawah ini.

Statechart diagram of an order management system



14 Pertemuan 15

14.1 Final Project

14.1.1 Review dan Pembahasan tugas besar penggunaan keseluruhan diagram dan view

DAFTAR PUSTAKA

Kenneth H. Rossen, Discrete Mathematics and Its Application, 4th edition, 1999, Mc-Graw Hill International

Bernard Kolman dan Robert C. Busby, Discrete Mathematical Structures For Computer Science, 2nd edition, 1987, Prentice Hall

C. L. Liu, Elements of Discrete Mathematics, 1992, Gramedia Pustaka Utama

Rinaldi Munir, Matematika Diskrit, Edisi 2, 2003, Penerbit Informatika Bandung

Jong Jek Siang, Matematika Diskrit Dan Aplikasinya Pada Ilmu Komputer, 2002, Penerbit Andi Yogyakarta