Pemodelan Sistem Perangkat Lunak

Budi susanto
FTI UKDW Yogyakarta

Modul #1

Pengantar Visual Modeling

Pengertian Visual Modeling

- *Visual Modeling*
  - *Cara berfikir terhadap permasalahan dengan menggunakan model-model yang diorganisasikan serupa dengan ide-ide pada dunia nyata* (--Terry Quatrani)
  - *Proses pengambilan informasi dari model dan menampilkannya secara grafis menggunakan sekumpulan elemen grafik standar* (-- Wendy and Michael Boggs)
- *Sebuah model adalah suatu penyederhanaan dari yang nyata*
  - Model menyediakan cetak biru dari suatu sistem

Manfaat Visual Modeling

- Memudahkan dalam memahami masalah
  - Kita membangun model agar kita dapat memahami sistem yang kita kembangkan secara lebih baik
  - Kita membangun model dari sistem yang komplek karena kita tidak dapat mengingat suatu sistem secara keseluruhan
- Mengkomunikasikan dengan setiap orang yang terlibat dalam proyek
- Memodelkan perusahaan
- Mempersiapkan dokumentasi
- Merancang program dan basis data

Prinsip Pemodelan

- Menurut Grady Booch, James Rumbaugh dan Ivar Jacobson*:*
  - *The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.*
  - *Every model may be expressed at different levels of precision.*
  - *The best models are connected to reality.*
  - *No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models.*

UML (Unified Modeling Language)

- UML adalah bahasa model standar untuk pengembangan cetak biru perangkat lunak.
- Bahasa model merupakan bahasa yang memiliki kamus kata dan aturan yang berpusat pada gambaran konseptual dan fisik dari suatu sistem
- UML sebagai bahasa model menyatakan bagaimana membuat dan membaca model dengan benar, namun tidak menyatakan model apa yang harus dibuat dan kapan seharusnya dibuat

Peran UML

UML adalah bahasa untuk

- Visualisasi
  - Menggambarkan ide dalam notasi dan semantik yang lebih mudah dipahami oleh siapapun
- Spesifikasi
  - spesifikasi dari semua keputusan penting analisa, perancangan, dan penerapan yang harus diambil dalam pengembangan dan deployment sistem PL

Peran UML (lanj)

- Konstruksi
  - UML bukan bahasa pemrograman visual
  - Model UML dapat dihubungkan secara langsung dengan beberapa bahasa pemrograman
  - *Forward engineering*: menghasilkan kode dari model
  - *Reverse engineering*: membangun model dari kode
- Dokumentasi
  - UML mencakup dokumentasi arsitektur sistem dan rincinya
  - Sebagai suatu bahasa untuk menyatakan kebutuhan dan pengujian.
  - UML menyediakan bahasa untuk aktifitas perencanaan proyek dan manajemen *release*

The Three Amigos

Sedikit Cerita tentang UML (1)

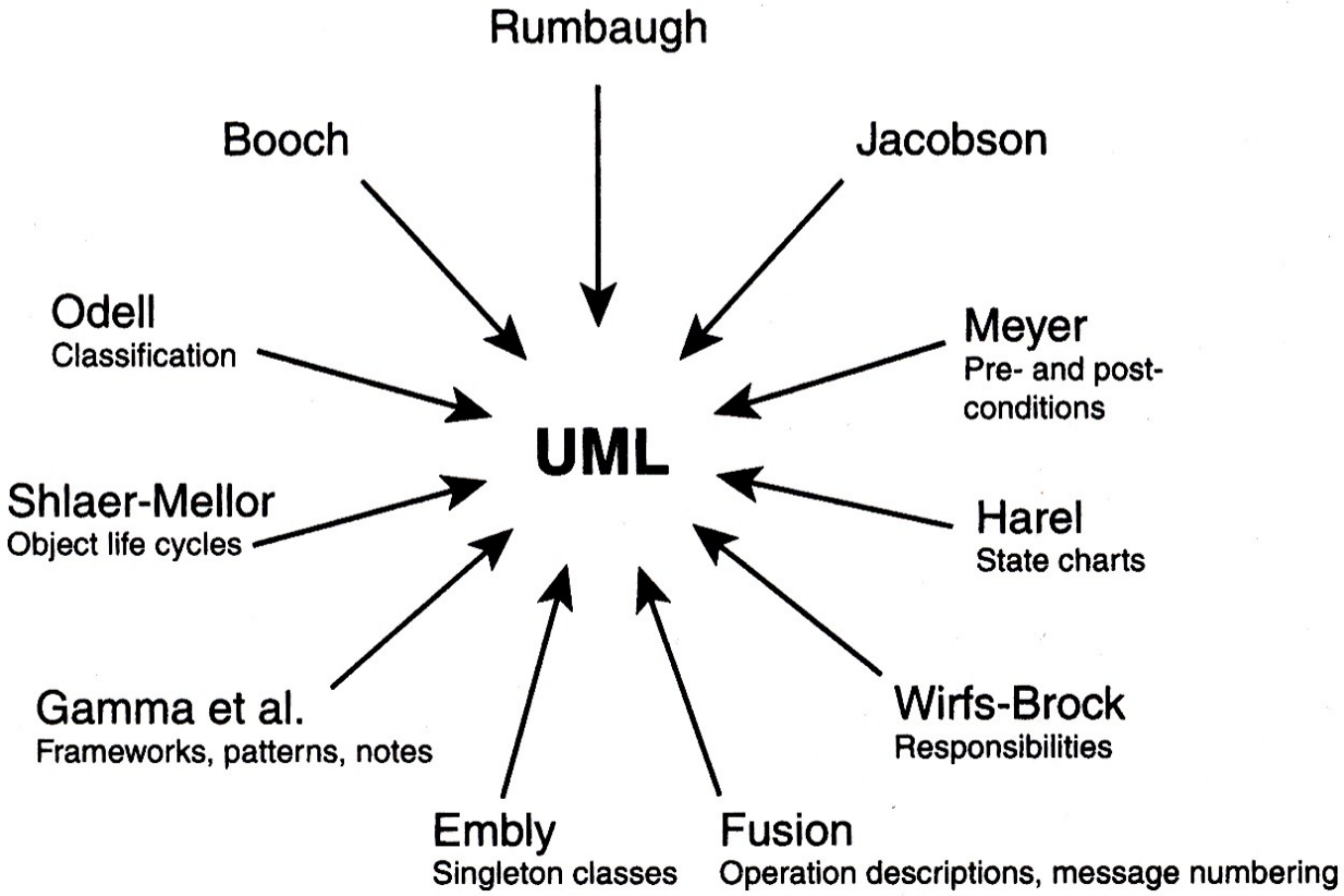
- 1980, Smalltalk lahir diikuti oleh C++
- 1998 – 1992, muncul beberapa notasi grafis untuk pemodelan OO
  - **Grady Booch [Booch, OOAD]**; Peter Coad [Coad, OOA], [Coad, OOD]; **Ivar Jacobson (Objectory) [Jacobson, OOSE]**; Jim Odell [Odell]; **Jim Rumbaugh (OMT) [Rumbaugh, insights], [Rumbaugh, OMT]**; Sally Shlaer and Steve Mellor [Shlaer and Mellor, data], [Shlaer and Mellor, states]; and Rebecca Wirfs-Brock (Responsibility Driven Design) [Wirfs-Brock].

Sedikit Cerita tentang UML (2)

- 1993, terjadi konsolidasi metodologi pemodelan OO (Jim Rumbaugh meninggalkan GE dan bergabung dengan Grady Booch di Rational)
- October 1995, Unified Method versi 0.8 (oleh Rational Software Co.)
- 1996, Unified Method diubah menjadi UML (setelah Ivar Jacobson bergabung - Three Amigos)
- July 1997, UML 1.0 disahkan dan diberikan ke OMG (Object Management Group)
- Sept 1997, OMG mengeluarkan UML 1.1 sebagai standar industri

Sedikit Cerita tentang UML (3)

- Release berikutnya 1.2, 1.3, 1.4 dan 1.5. Terakhir adalah 2.0



Blok Pembangun UML

- *Thing*
  - "penghuni" paling elit dalam model
    - *Structural, Behavioral, Grouping, Annotations*
- *Relationship*
  - Menghubungkan antar *thing*
    - *Dependency, Association, Generalization, Realization*
- *Diagram*
  - Kumpulan *thing* dan *relationship*
    - *Lihat tipe giagram UML*

Structural Things (1)

- Structural thing merupakan kata benda model UML
  - **Class**
    - Gambaran dari sekumpulan objek yang berbagi atribut, operasi, hubungan dan semantik yang sama
    - Sebuah class menerapkan satu atau lebih antarmuka
  - **Interface**
    - Sekumpulan operasi yang menyatakan layanan suatu class atau component
    - Suatu interface menggambarkan perilaku objek yang disediakan untuk pihak luar.
    - Interface bisa saja menyatakan sebagian atau seluruh perilaku suatu class/component.

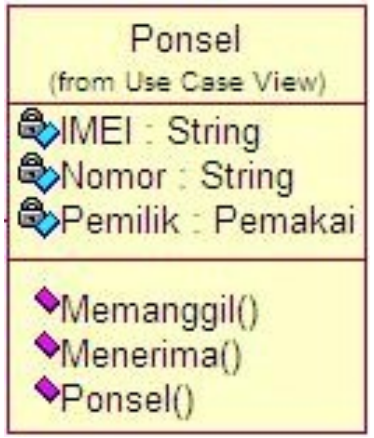
Structural Things (2)

- **Collaboration**
  - Suatu interaksi dan suatu kumpulan aturan dan elemen lain yang bekerja bersama untuk menyediakan perilaku yang bekerja sama
  - Menyatakan penerapan pola yang membentuk sistem
- **Use Case**
  - Suatu gambaran dari sekumpulan urutan aksi yang terjadi pada sistem yang menghasilkan suatu nilai kepada actor
  - Digunakan untuk menstrkturkan perilaku sistem dalam suatu model
  - Merupakan realisasi dari collaboration.

Structural Things (3)

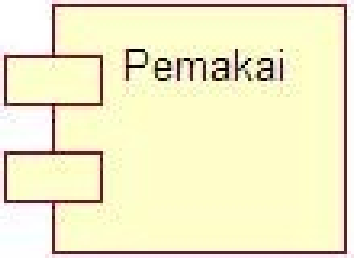
- **Active Class**
  - Suatu class dimana objek memiliki satu atau lebih proses/thread sehingga dapat memulai aktifitas kontrolnya
- **Component**
  - Suatu bagian fisik dan *replaceable* dari suatu sistem menyediakan realisasi dari sekumpulan antarmuka
  - Menyatakan paket fisik daripada elemen logika, seperti class, interface dan collaboration
- **Node**
  - Elemen fisik yang ada pada saat run time dan menyatakan suatu sesumber perhitungan, yang memiliki memori dan kemampuan pemrosesan

Simbol-simbol Structural Things



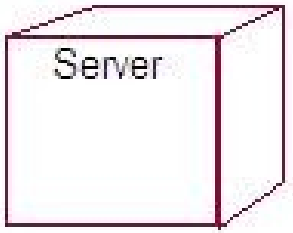
Class

Interface



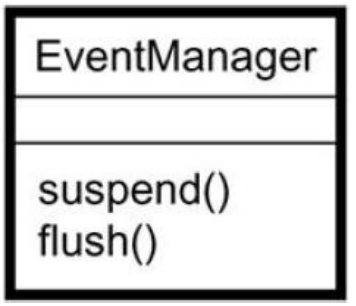
Component

Collaboration



Node

Use Case

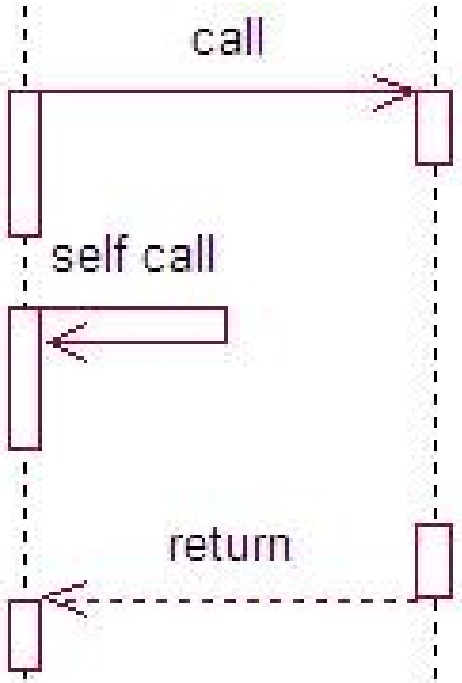


Active Class

Behavioral Things (1)

- Behavioral things merupakan bagian dinamis dari model UML, yang menyatakan kata kerja dari suatu model, menyatakan perilaku sepanjang waktu dan ruang.
  - *Interaction:* suatu perilaku yang terdiri dari sekumpulan pertukaran pesan diantara sekumpulan objek dalam suatu kontek untuk mencapai suatu tujuan
  - *state machine:* suatu perilaku yang menentukan urutan *state* dari suatu objek atau interaksi terhadap objek tersebut selama “hidup”nya.

Simbol-simbol Behavioral Things

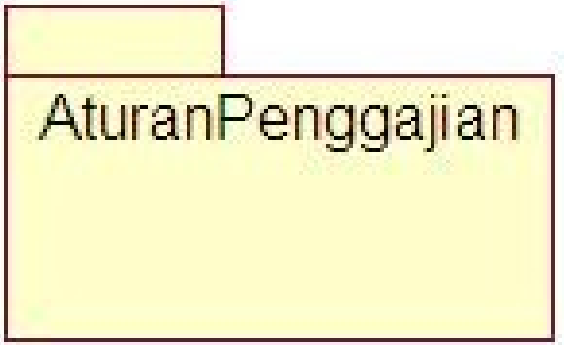


message

State Machine

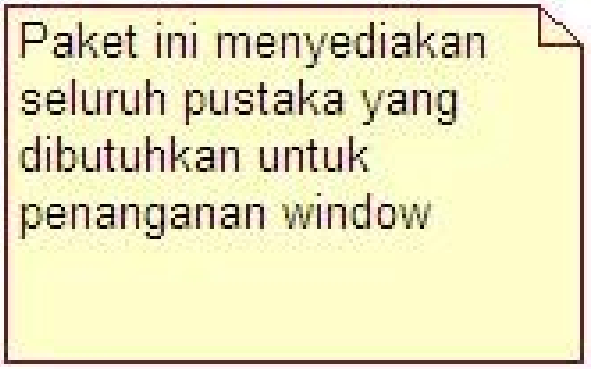
Grouping Things

- *Grouping things* : bagian organisasional model UML.
- *Package* : mekanisme umum untuk mengorganisasikan elemen-elemen ke dalam kelompok-kelompok
  - *Structural things*, *behavioral things*, dan pengelompokan yang lain dapat ditempatkan dalam sebuah package.
  - Berbeda dengan component (yang ada selama *run time*), package sepenuhnya konseptual (ada hanya pada waktu pengembangan)



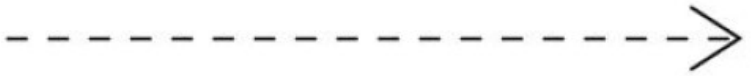
Annotations Things

- *Annotational things* : bagian model UML yang memberikan keterangan.
  - *Note*: suatu simbol yang memberikan batasan dan komentar yang dikaitkan pada suatu elemen atau kumpulan elemen

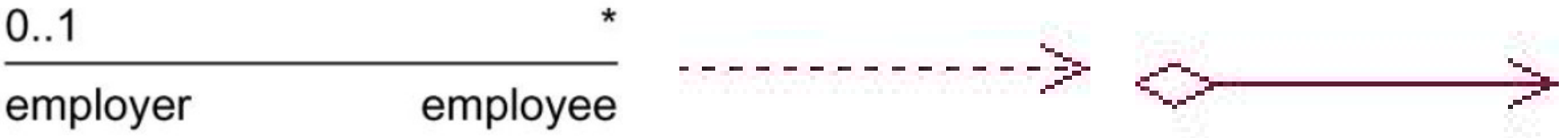


Relationships (1)

- **Dependency**
  - Suatu hubungan semantik antara dua *things* dimana perubahan pada satu *thing* (*independent*) mungkin mempengaruhi semantik *thing* (*dependent*) lain.



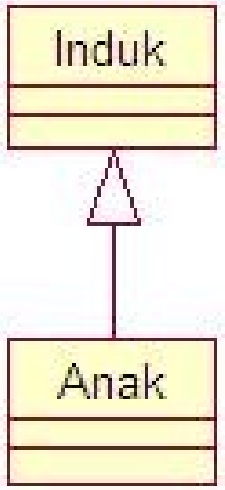
- **Association**
  - Hubungan struktural yang menggambarkan sehimpunan mata rantai antar objek.



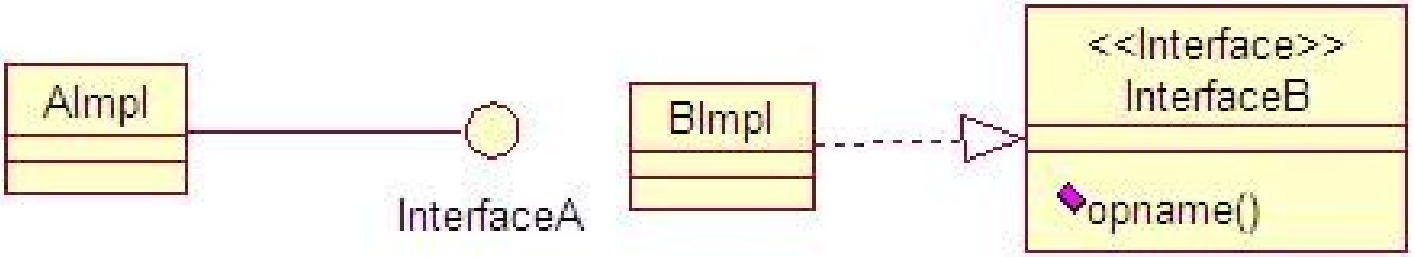
  - *Aggregation* merupakan bentuk association khusus yang menyatakan hubungan struktural antara *whole* dan *part*

Relationships (2)

- **Generalization**
  - Hubungan spesialisasi dimana objek dari elemen khusus (anak) merupakan pengganti untuk objek elemen umum (induk)

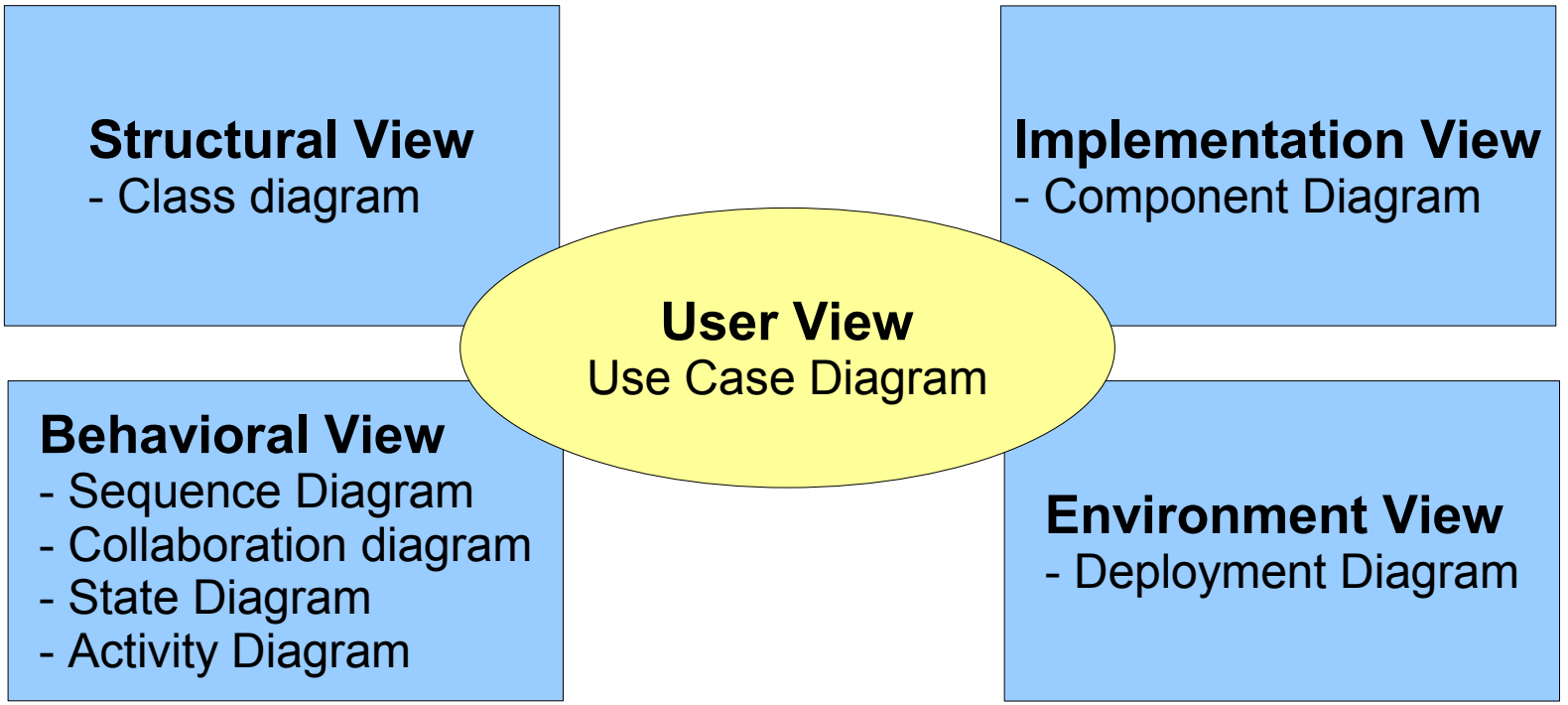


- **Realization**
  - Hubungan antara antarmuka yang tersedia secara umum (interface atau use case) dengan penerapan detail dari antarmuka (class, package, atau use case realization).

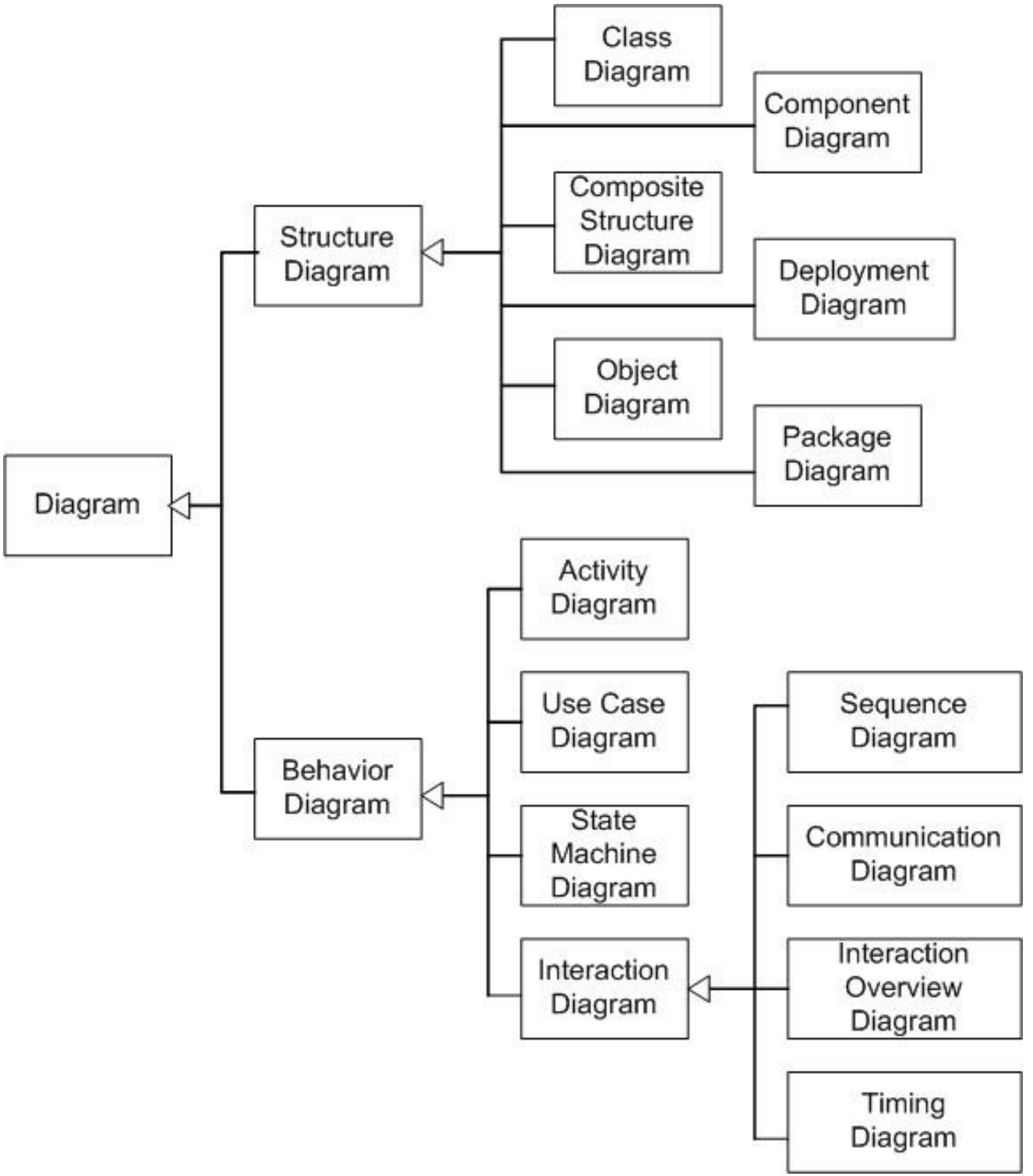


UML View

- Arsitektur dan perancangan sistem memungkinkan kita untuk mengatur cara pandang yang berbeda terhadap sistem dan kontrol terhadap sistem keseluruhan
- UML memungkinkan kita untuk memodelkan sistem untuk kepentingan berbagai cara pandang.



UML 2.0 Diagram



Tipe Diagram UML 2.0

Diagram	Tujuan	Versi
Activity	Perilaku prosedural dan paralel	UML 1
Class	Class, karakteristik, dan hubungan	UML 1
Communication	Interaksi antar objek; ditekankan hubungannya	UML 1 collaboration diagram
Component	Struktur Komponen dan hubungan	UML 1
Composite structure	penguraian runtime suatu class	UML 2
Deployment	Deployment pada node	UML 1
Interaction overview	Gabungan diagram sequence dan activity	UML 2
Object	Contoh konfigurasi instance	UML 1 (tidak resmi)
Package	Struktur hirarki Compile-time	UML 1 (tidak resmi)
Sequence	Interaksi antar objek; ditekankan pada urutan	UML 1
State machine	Bagaimana kejadian mengubah objek sepanjang hidupnya	UML 1
Timing	Interaksi antar objek; ditekankan pada timing	UML 2

Diagram-diagram UML (1)

- **Business Use Case Diagrams**
  - Digunakan untuk menyatakan fungsionalitas yang disediakan oleh suatu organisasi secara keseluruhan
  - Digunakan secara intensif selama aktifitas pemodelan bisnis untuk menghimpun kontek sistem dan membentuk dasar pembuatan use case
  - Tidak dibedakan antara proses manual atau otomatis (use case fokus ke proses otomatis) (dipandang dari sudut pandang organisasi)
  - *Business use case*: proses fungsional bisnis
  - *Business actor*: peran dimana bisnis berinteraksi

Diagram-diagram UML (2)

- **Business Use Case Diagrams**
  - Kapan digunakan?
    - Merupakan organisasi yang baru
    - Sedang menjalankan dan atau sedang menerapkan rekayasa ulang bisnis
    - Membangun software yang berimbas pada bagian penting perusahaan
    - Terdapat workflow komplek yang tidak terdokumentasi
    - Konsultan pada organisasi yang baru
  - Kapan tidak perlu membuat Business Use Case?
    - Sudah memahami visi, misi dan workflow organisasi
    - Hanya membangun software yang tidak berimbas besar pada organisasi
    - Jika tidak tersedianya waktu.

Contoh Business Use Case

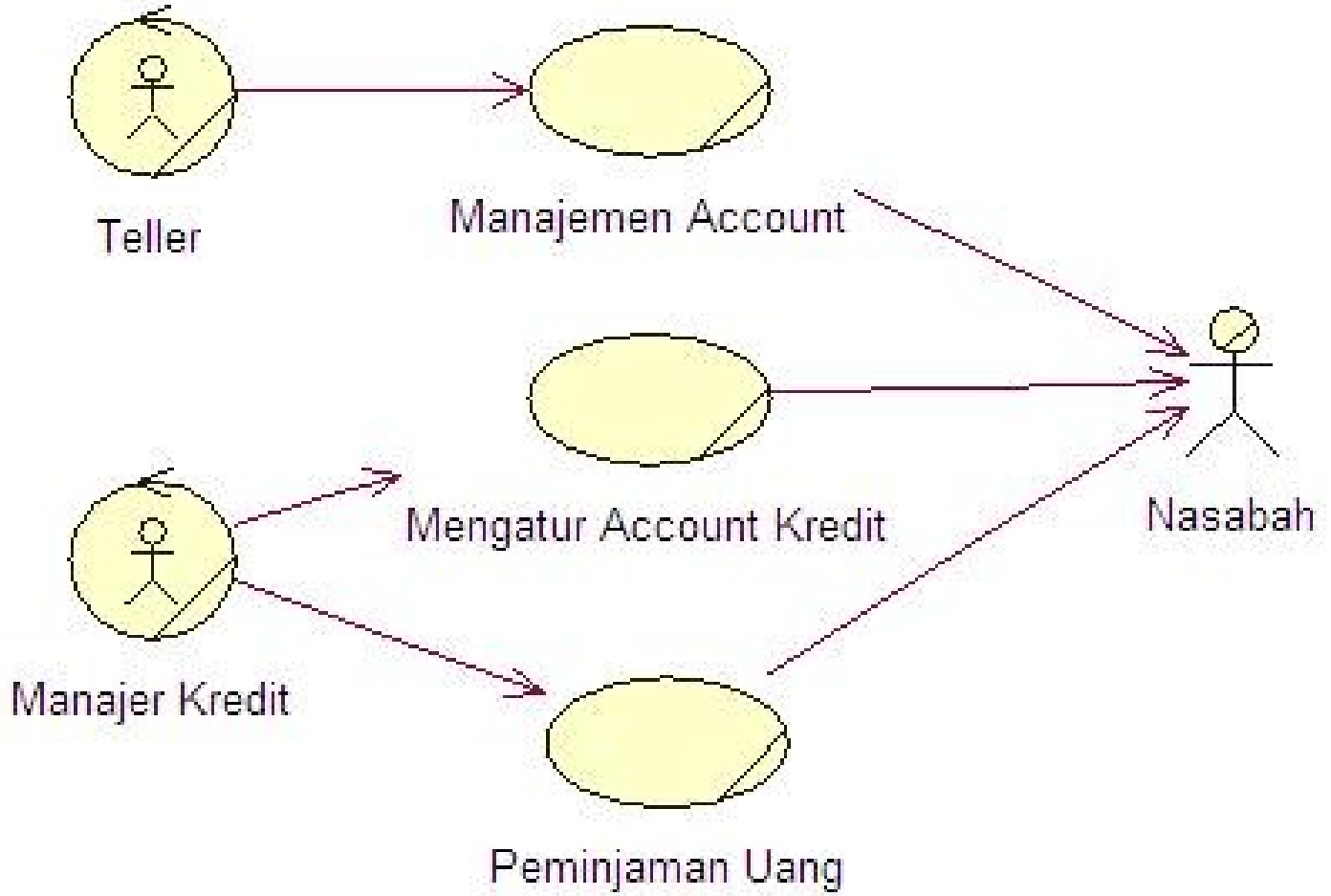


Diagram-diagram UML (3)

- **Use Case Diagrams**
  - Memperlihatkan inetarksi antara use case dan actor.
  - Use cases mewakili fungsionalitas sistem, kebutuhan sistem dari sudut pandang pemakai.
  - Actors mewakili orang atau sistem yang menyediakan atau menerima informasi dari sistem.

Contoh Use Case

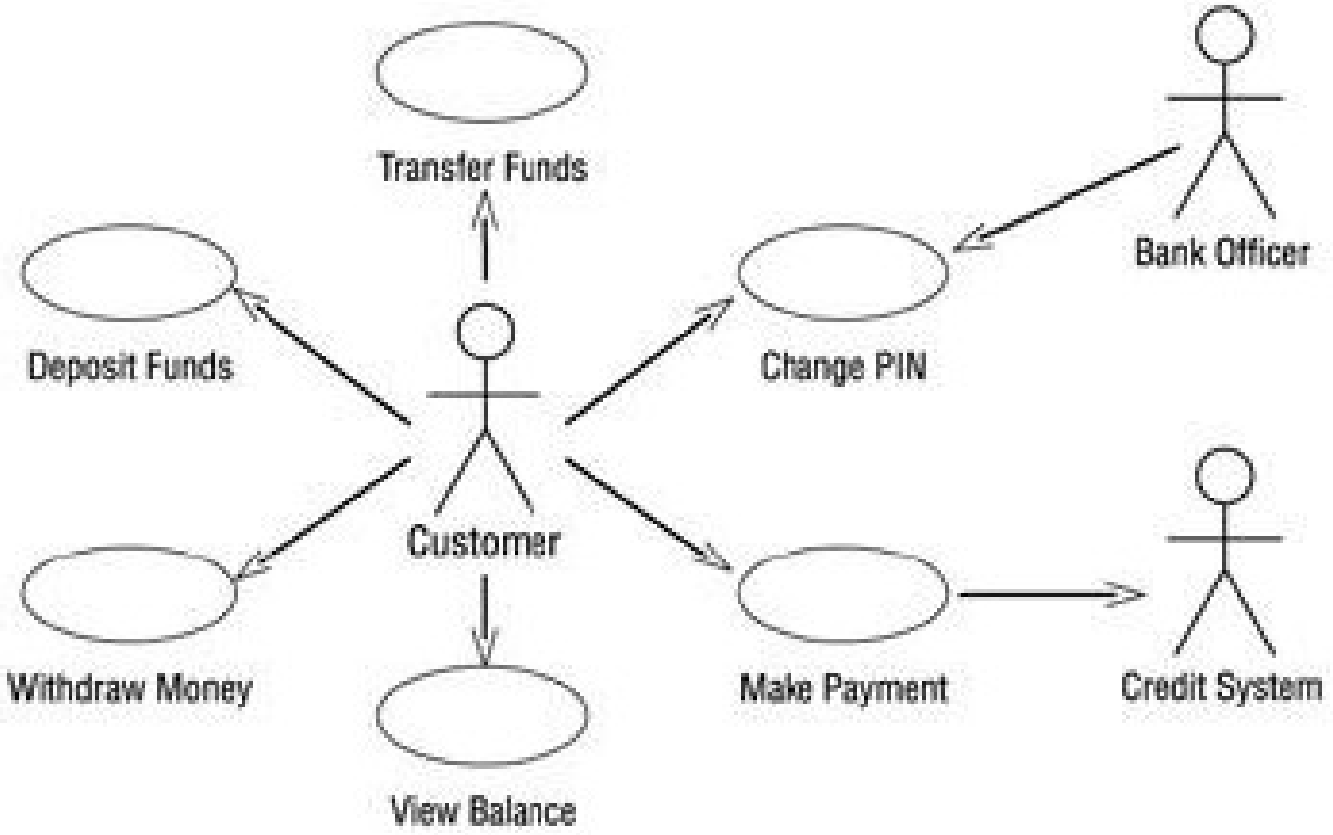


Diagram-diagram UML (4)

- **Activity Diagram**
  - Menggambarkan aliran fungsionalitas dalam suatu sistem.
  - Dapat digunakan dalam pemodelan bisnis untuk menunjukkan *business workflow*.
  - Atau juga digunakan dalam analisa kebutuhan untuk menggambarkan aliran kejadian melalui suatu use case.
  - Mendefinisikan dimana workflow dimulai, dimana berhentinya, aktifitas apa yang terjadi selama workflow, bagaimana urutan kejadian aktifitas
  - Suatu aktifitas adalah suatu pekerjaan yang dilaksanakan selama workglow

Contoh Activity Diagram

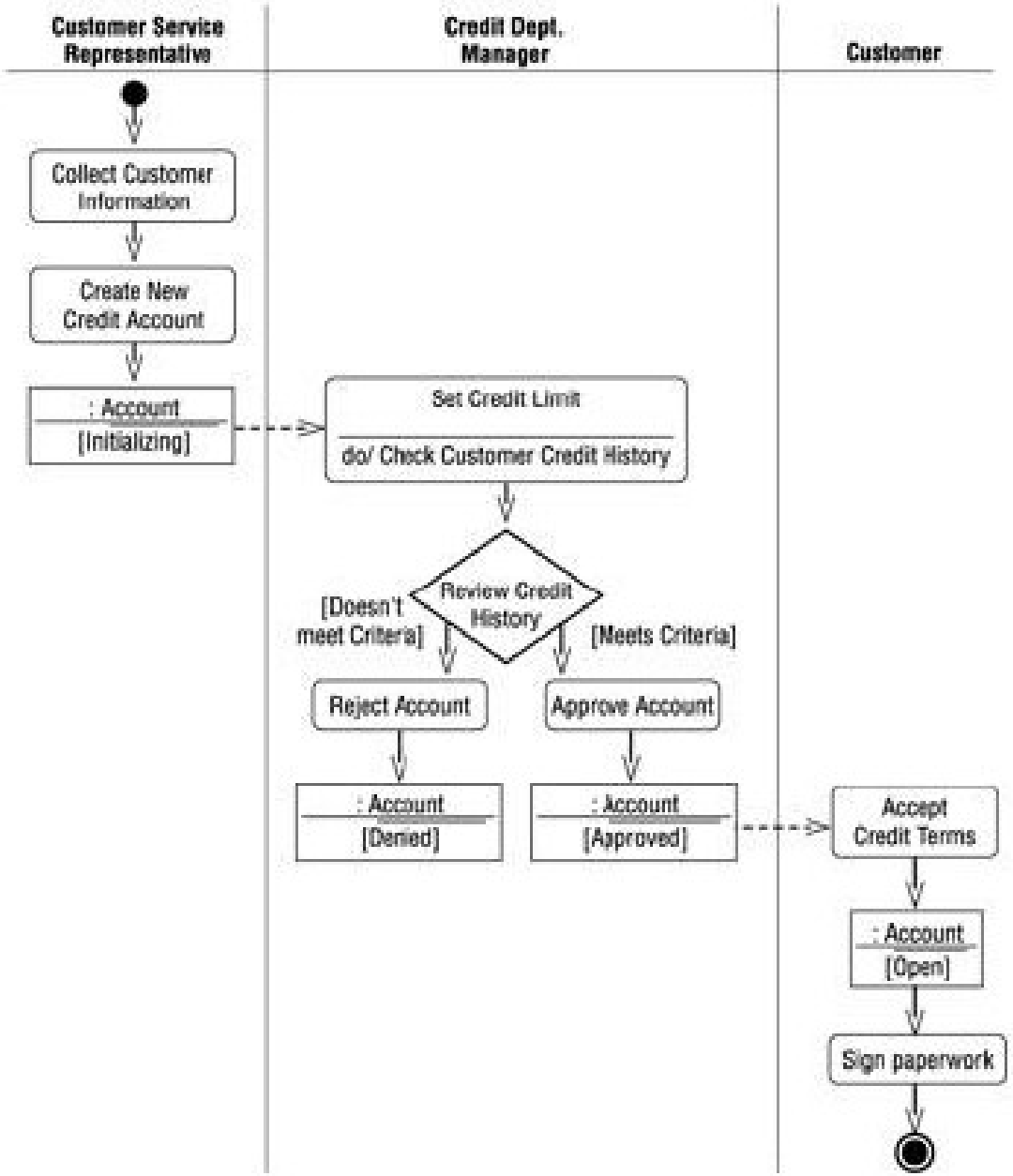
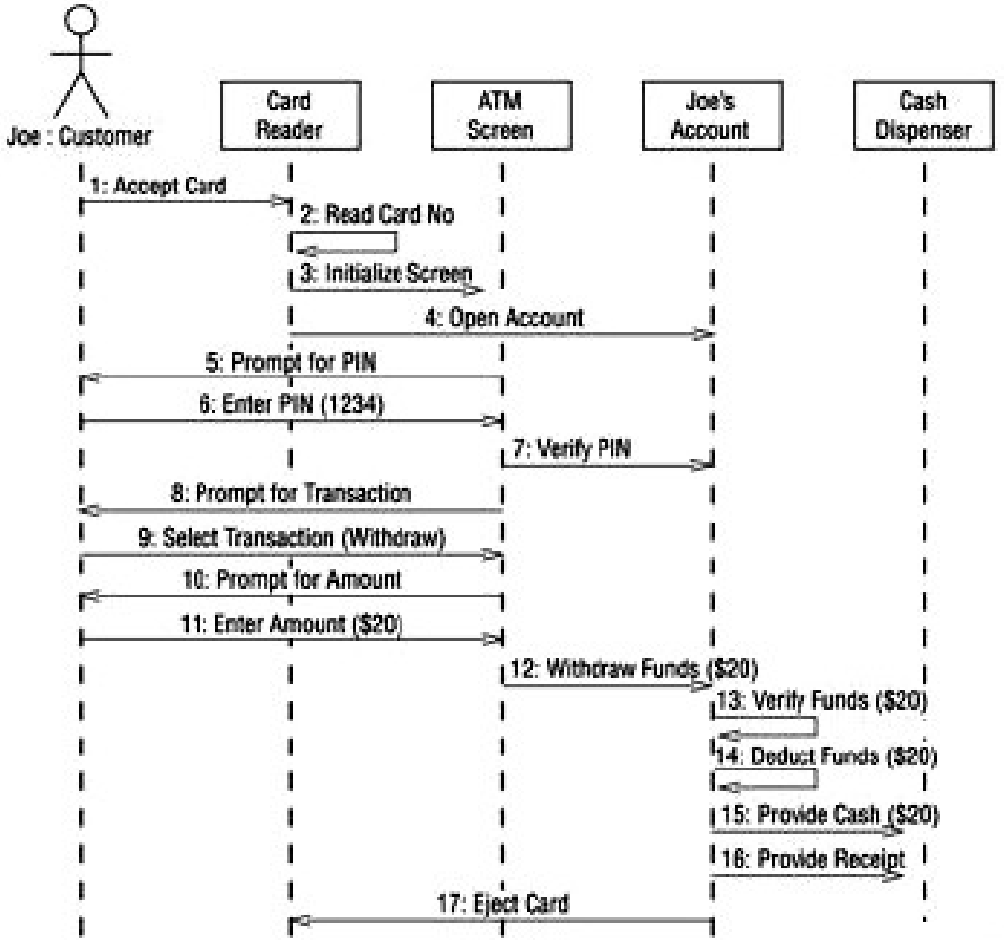


Diagram-diagram UML (5)

- **Sequence Diagram**
  - Merupakan diagram interaksi yang menekankan urutan waktu pertukaran pesan
- **Collaboration Diagram**
  - Collaboration diagrams menampilkan informasi yang sama dengan Sequence diagrams.
  - Hanya saja, interaksi antara objek dan actor tidak ditampilkan berdasar urutan waktu.

Contoh Sequence Diagram dan Collaboration Diagram



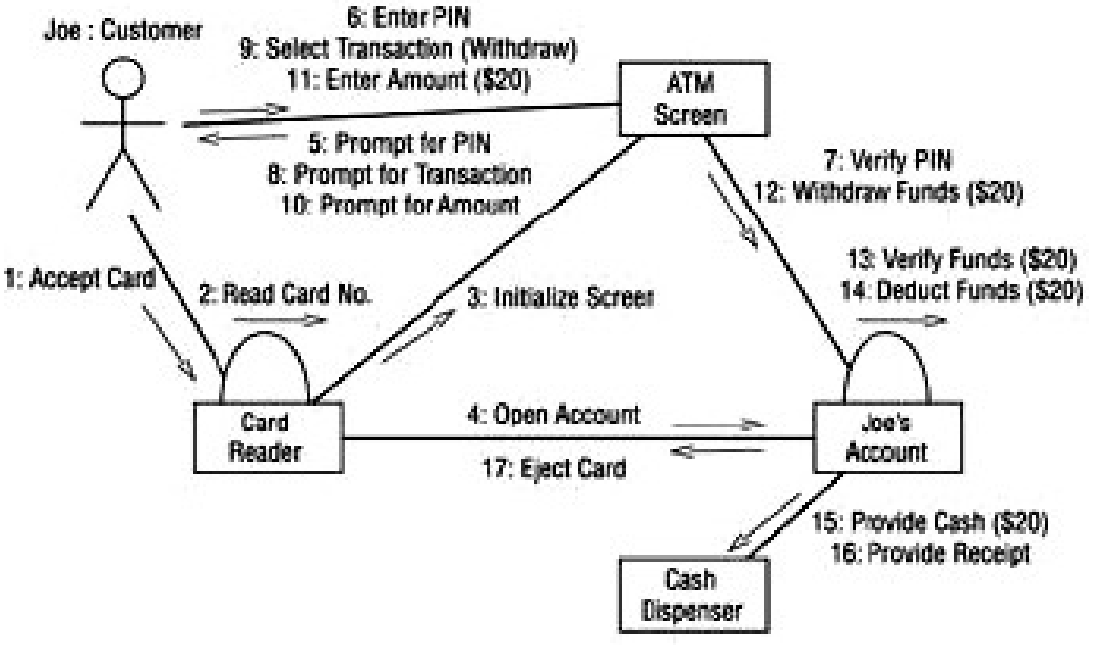


Diagram-diagram UML (6)

- **Class Diagram**
  - Memperlihatkan interaksi antar class dalam sistem.
  - Class merupakan suatu cetak biru untuk objek
  - Memperlihatkan gambaran statik dari class-class dan hubungannya
- **Statechart Diagram**
  - Menyediakan cara memodelkan berbagai state keberadaan objek.
  - Digunakan untuk memodelkan lebih dinamis perilaku dari sistem

Contoh Class dan Statechart Diagram

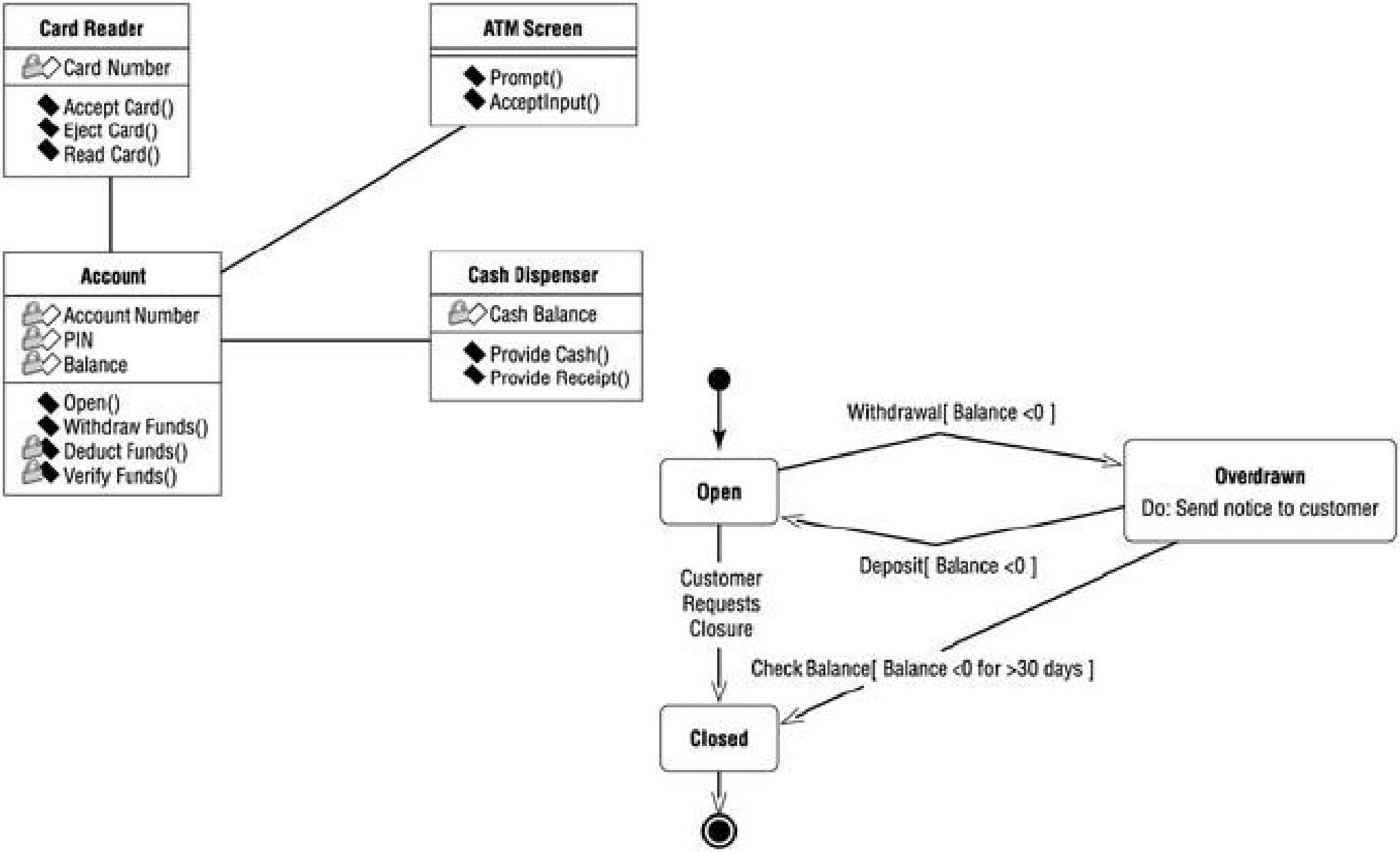
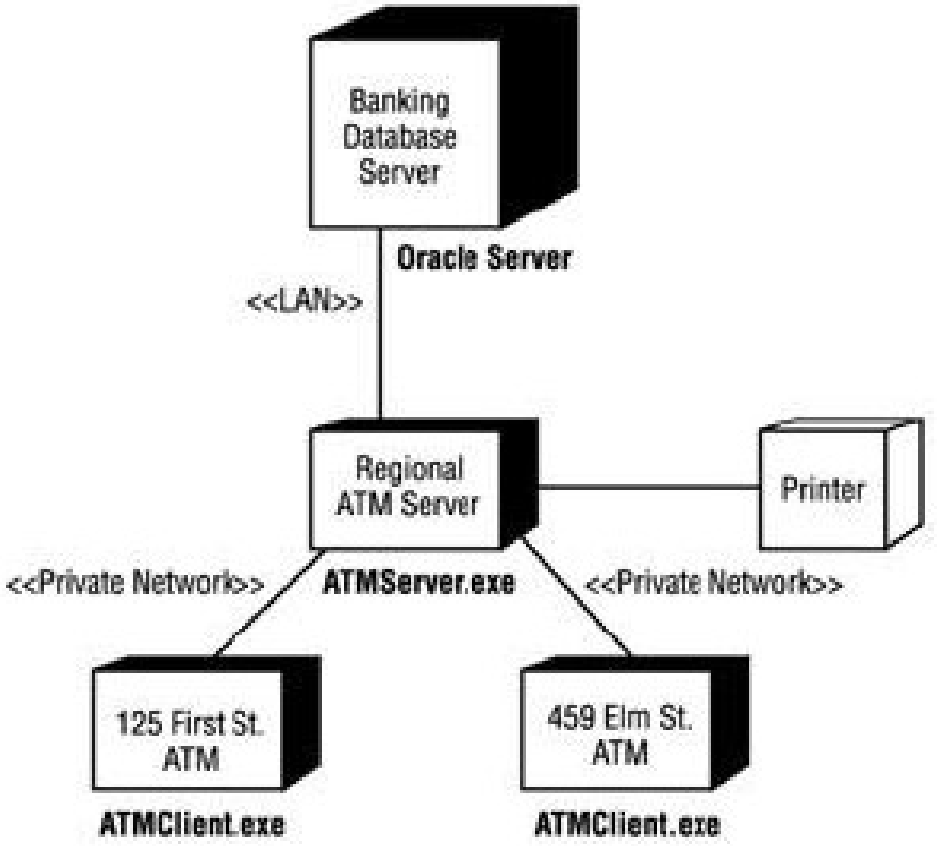


Diagram-diagram UML (7)

- **Component Diagrams**
  - Memperlihatkan organisasi dan ketergantungan antara sekumpulan komponen. (Implementation view)
  - Dihubungkan dengan class diagram dimana komponen tersebut memetakan satu atau lebih class, interface, atau collaboration.
- **Deployment Diagrams**
  - Memperlihatkan konfigurasi run-time pada node pemrosesan dan komponen yang berjalan padanya.
  - *(static deployment* view)
  - Dihubungkan dengan component diagrams dimana sebuah node menyertakan satu atau lebih komponen

Contoh Component dan Deployment Diagram



Pemakai model UML (1)

- Tidak hanya pengembang yang akan menggunakan UML
  - Seluruh tim akan menggunakan Business Use Case diagram untuk memahami bisnis dalam sistem.
  - Pelanggan dan manajer proyek akan menggunakan Use Case diagram untuk mendapat gambaran level tinggi dari sistem dan menyetujui ruang lingkup proyek.
  - Manajer proyek akan menggunakan Use Case diagram dan dokumentasi untuk membagi proyek ke dalam sub proyek yang mudah diatur.

Pemakai model UML (2)

- Analis dan pelanggan akan meninjau dokumentasi use case untuk melihat fungsi apa saja yang akan disediakan oleh sistem.
- Penulis teknis akan melihat pada dokumentasi use case untuk memulai menulis user manual dan rencana pelatihan.
- Analis dan pengembang akan melihat Sequence dan Collaboration diagram untuk mendapat gambaran bagaimana aliran logika dalam sistem, objek dalam sistem dan pertukaran pesan antar objek.
- Pegawai asuransi kualitas (QA) menggunakan dokumentasi use case dan Collaboration diagram untuk mendapatkan informasi yang mereka butuhkan untuk menguji script.

Pemakai model UML (3)

- Pengembang menggunakan Class dan Statechart diagram untuk mendapat gambaran detil sebagian sistem dan bagaimana mereka menghubungkan.
- Pegawai Deployment menggunakan Component dan Deployment diagram untuk melihat file executable, DLL , atau komponen lain yang akan dibuat, dan dimana komponen tersebut akan di-*deploy* pada jaringan.
- Seluruh tim menggunakan model untuk memastikan kebutuhan terselusuri ke kode program, demikian juga sebaliknya.

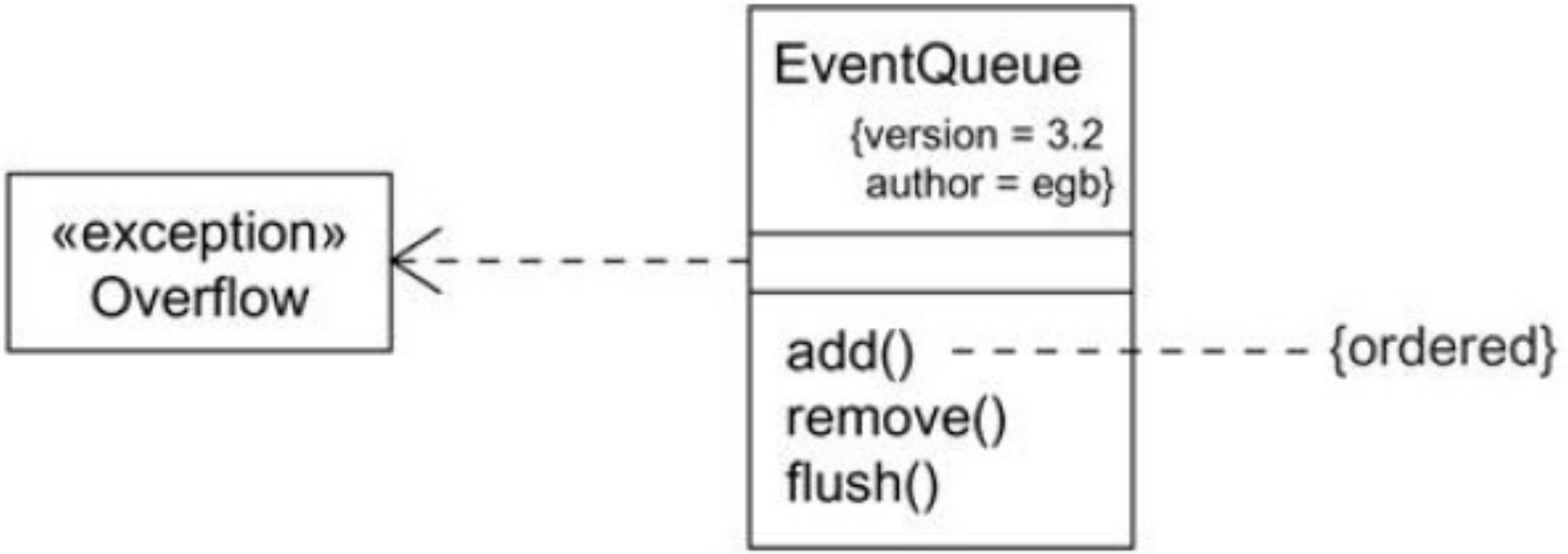
Mekanisme Umum dalam UML (1)

- ***Specification***
  - Setiap notasi grafik UML memiliki spefisikasi yang merupakan pernyataan sintak dan semantiknya
- ***Adornment***
  - Setiap elemen dalam UML memiliki notasi grafik unik yang menyediakan visualisasi aspek penting dari elemen yang bersangkutan
- ***Common Division***
  - Pembagian antara class dan object
  - Pemisahan antara interface dan implementasinya

Mekanisme Umum dalam UML (2)

- ***Extensibility Mechanism***
  - **Stereotype**
    - Memperluas kamus kata UML, yang memungkinkan kita membangun blok yang diturunkan dari yang sudah ada namun khusus untuk masalah yang ada. Contoh: Exception pada Java/C++ yang sebetulnya sebuah class yang memiliki fungsi khusus.
  - **Tagged Value**
    - Memperluas informasi dari spesifikasi elemen
  - **Constraint**
    - Memperluas semantik dari blok UML yang dibangun

Contoh Perluasan Mekanisme (3)



- (*Stereotype*) Class Overflow sebagai class khusus exception
- (*tagged value*) Class EventQueue yang digunakan adalah versi 3.2 yang dibuat oleh egb
- (*constraint*) semua operasi add() harus menjaga urutan antriannya