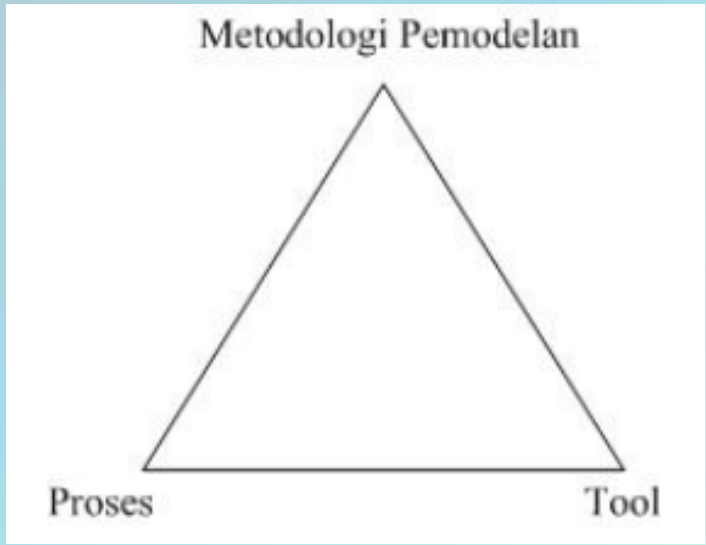
PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan 11

UML

Pemodelan

- Pemodelan (modeling) adalah proses merancang piranti lunak sebelum melakukan pengkodean (coding). Model piranti dapat dianalogikan seperti pembuatan blueprint pada pembangunan gedung.
- Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yaitu: metode pemodelan (notation), proses (process) dan tool yang digunakan.



UML

- UML (Unified Modelling Language) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan system piranti lunak.
- Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak.
- Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan.

Konsep Dasar UML

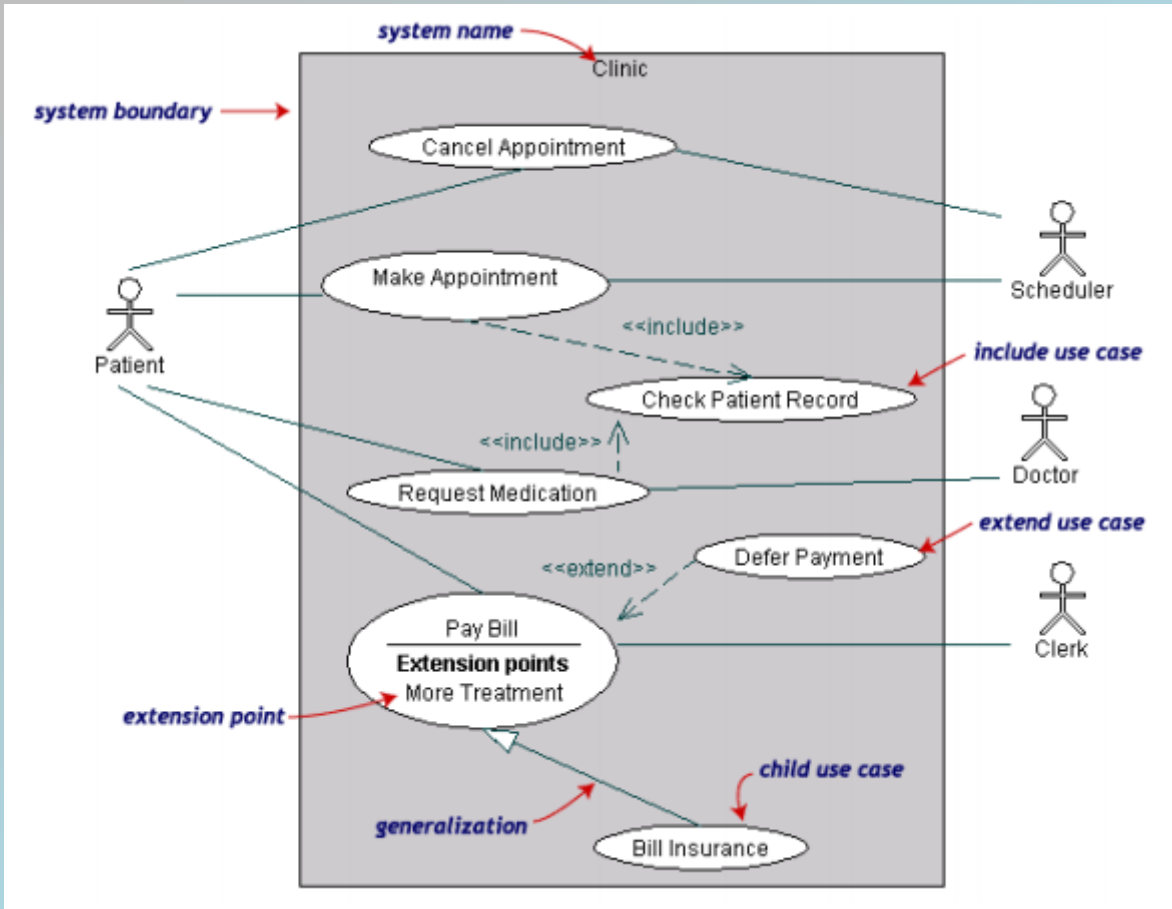
Berikut konsep dasar UML:

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Use Case Diagram

- Use Case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem.
- Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
- Sebuah use case merepresentasikan sebuah interaksi antara actor dengan sistem.
- Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dsb.
- Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

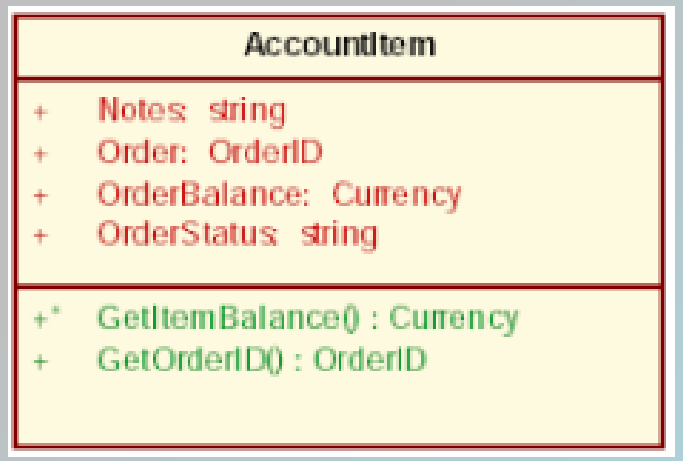
Contoh Use Case Diagram

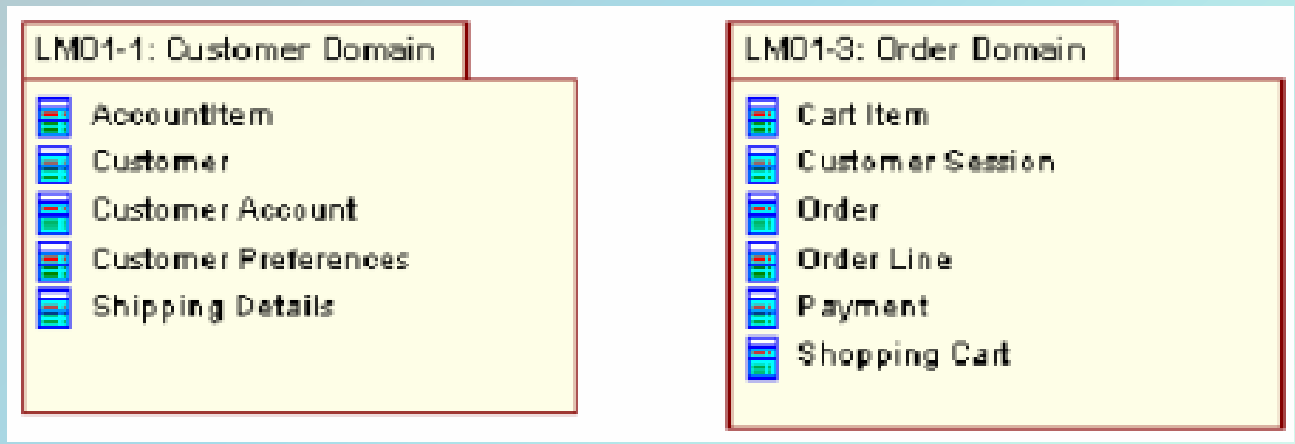


Class Diagram

- Class adalah sebuah spesifikasi yang jika diinstanisasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.
- Class menggambarkan keadaan (atribut/property) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi).
- Class diagram menggambarkan struktur dan deskripsi class, package, dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dll.
- Class memiliki tiga bagian utama, yaitu: nama, atribut, dan metode.

Class Diagram

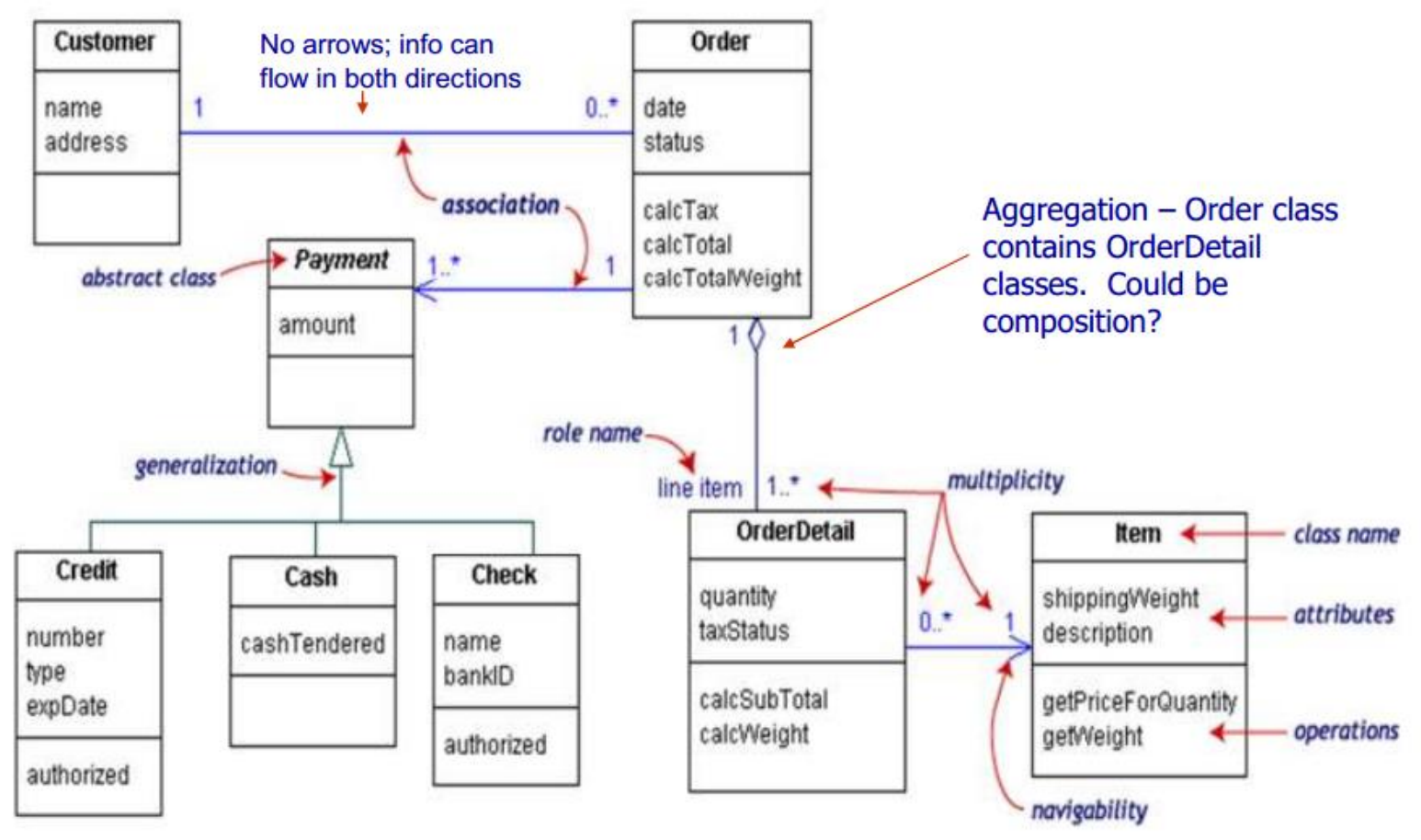




Hubungan Antar Kelas

1. Asosiasi, yaitu hubungan statis antar kelas. Umumnya menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui eksistensi kelas lain. Panah navigability menunjukkan arah query antar kelas.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas”).
3. Pewarisan, yaitu hubungan hirarkis antar kelas. Kelas dapat diturunkan dari kelas lain dan mewarisi semua atribut dan metode kelas asalnya dan menambahkan fungsionalitas baru, sehingga disebut anak dari kelas yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu kelas kepada kelas lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram.

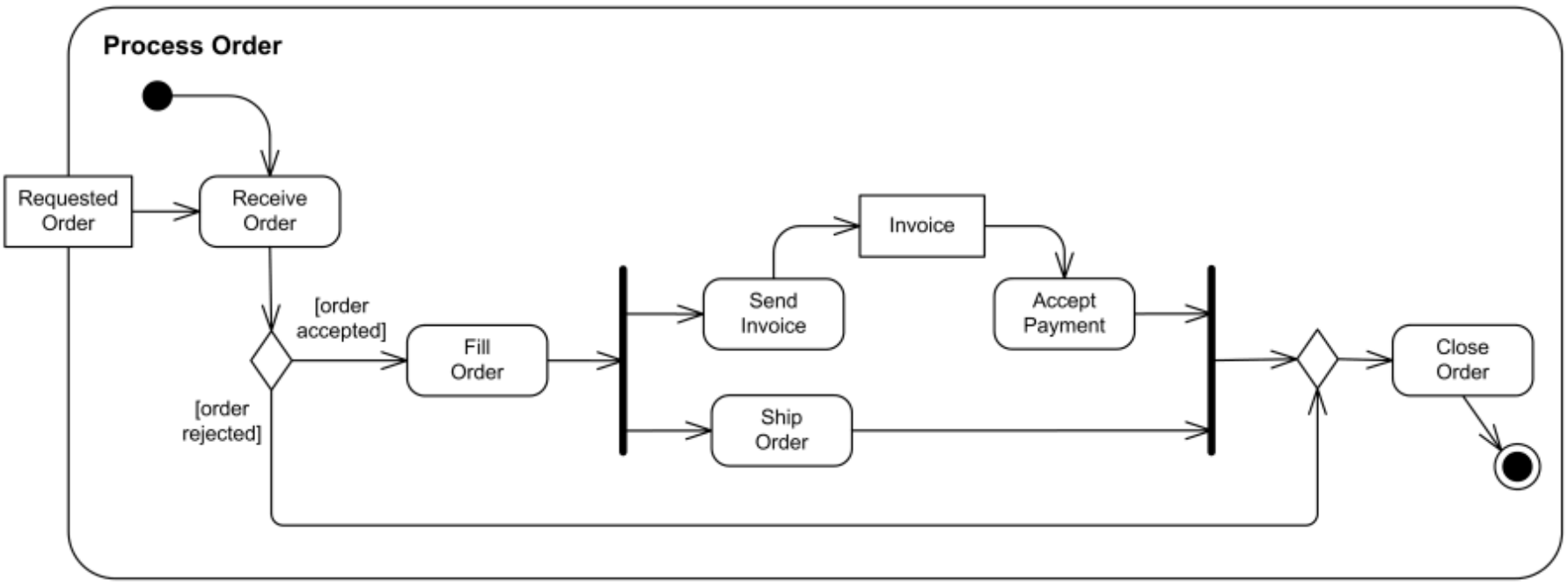
Contoh Class Diagram



Statechart Diagram

- Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statediagram menggambarkan kelas tertentu (satu kelas dapat memiliki lebih dari satu statechart diagram).
- State digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu.
- Transisi antar state umumnya memiliki kondisi guard yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku.
- Action yang dilakukan sebagai akibat dari event tertentu dituliskan dengan diawali garis miring.
- Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

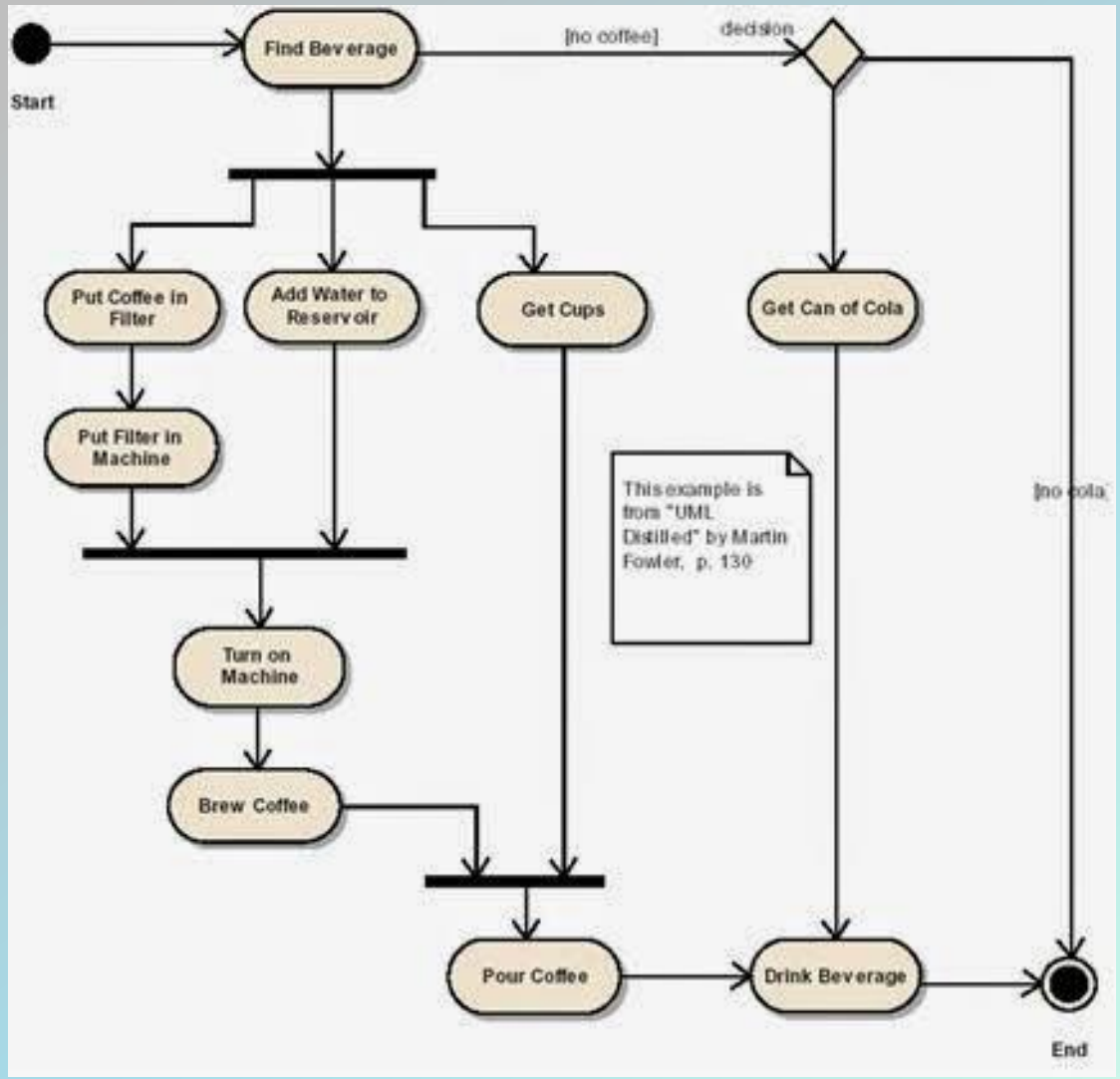
Contoh Statechart Diagram



Activity Diagram

- Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.
- Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*).
- Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Contoh Activity Diagram



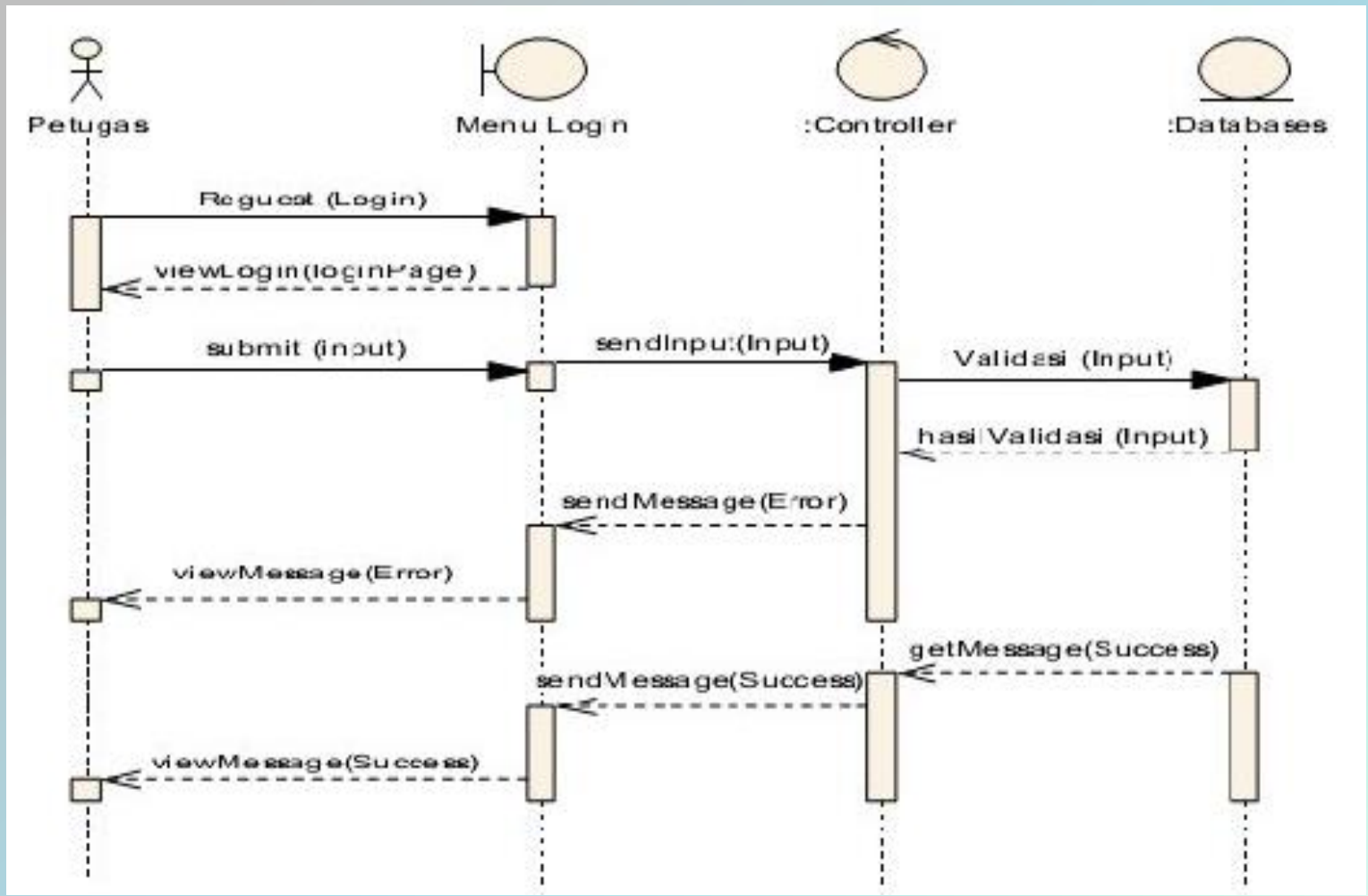
Sequence Diagram

- *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dsb) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait).
- *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Sequence Diagram (2)

- Masing-masing objek, termasuk actor memiliki lifeline vertical.
- *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya.
- Pada fase desain berikutnya *message* akan dipetakan menjadi operasi/metode dari kelas.
- *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

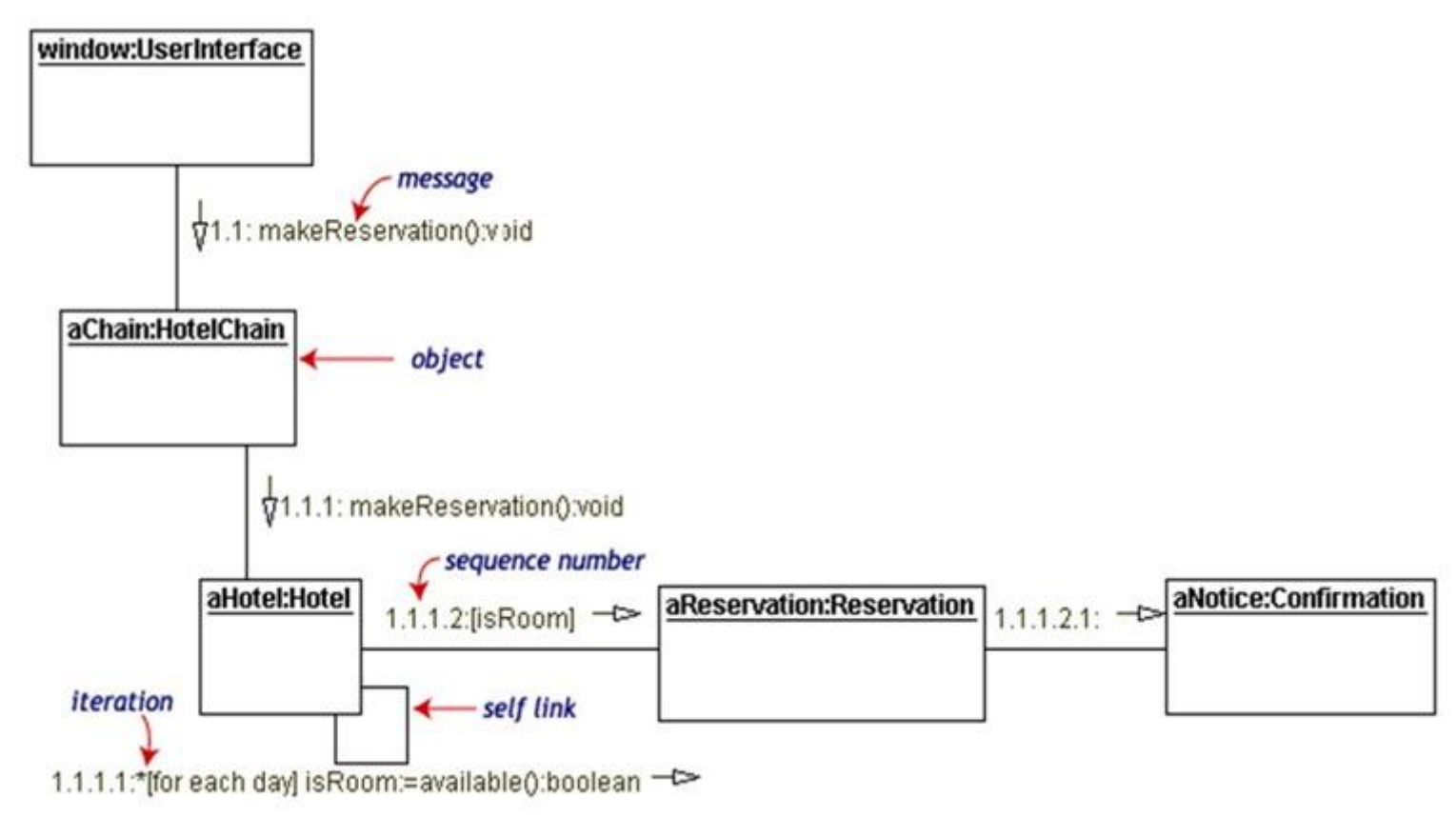
Contoh Sequence Diagram



Collaboration Diagram

- *Collaboration diagram* menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.
- Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefix yang sama.

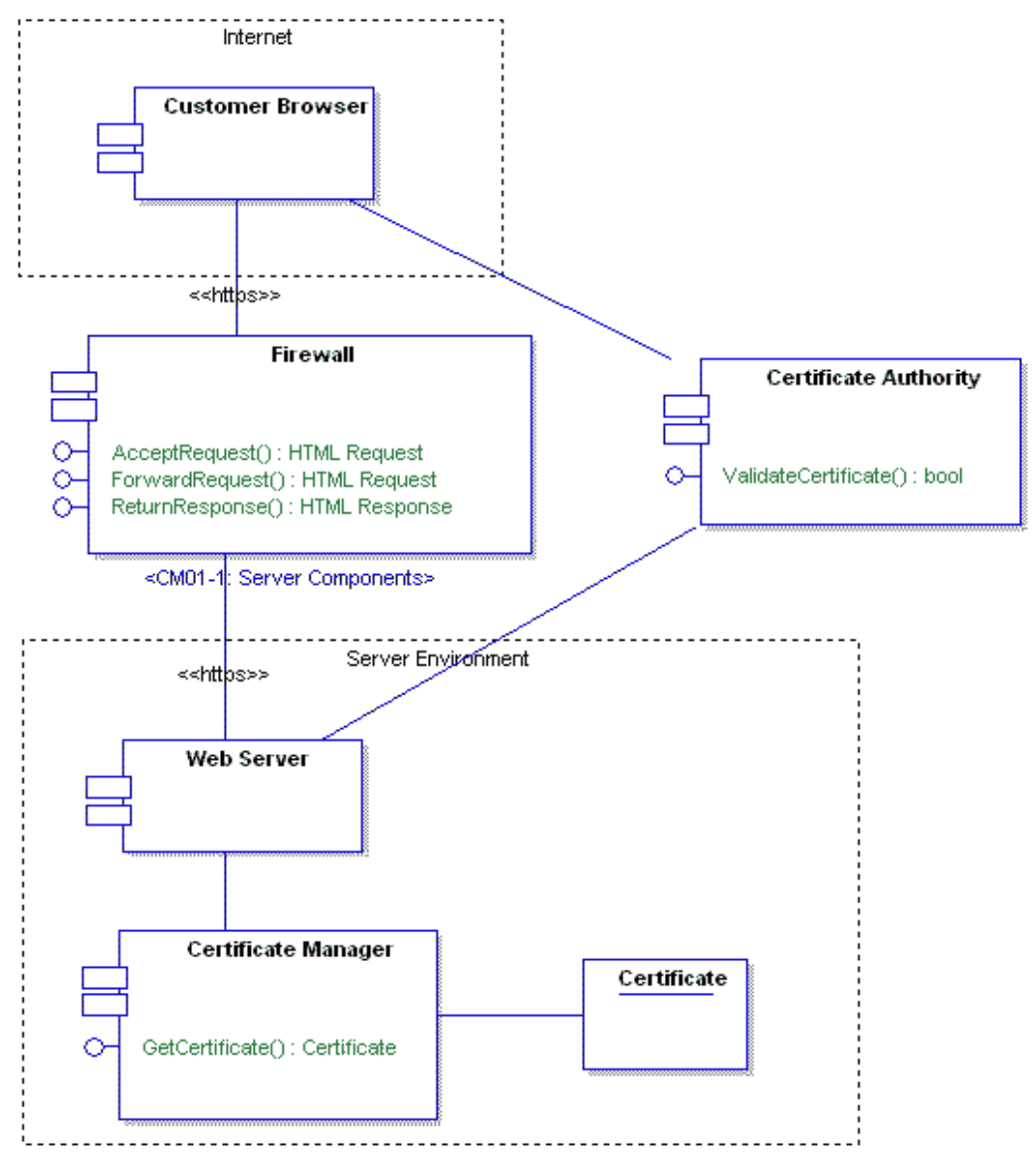
Contoh Collaboration Diagram



Component Diagram

- *Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.
- Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*.
- Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.
- Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

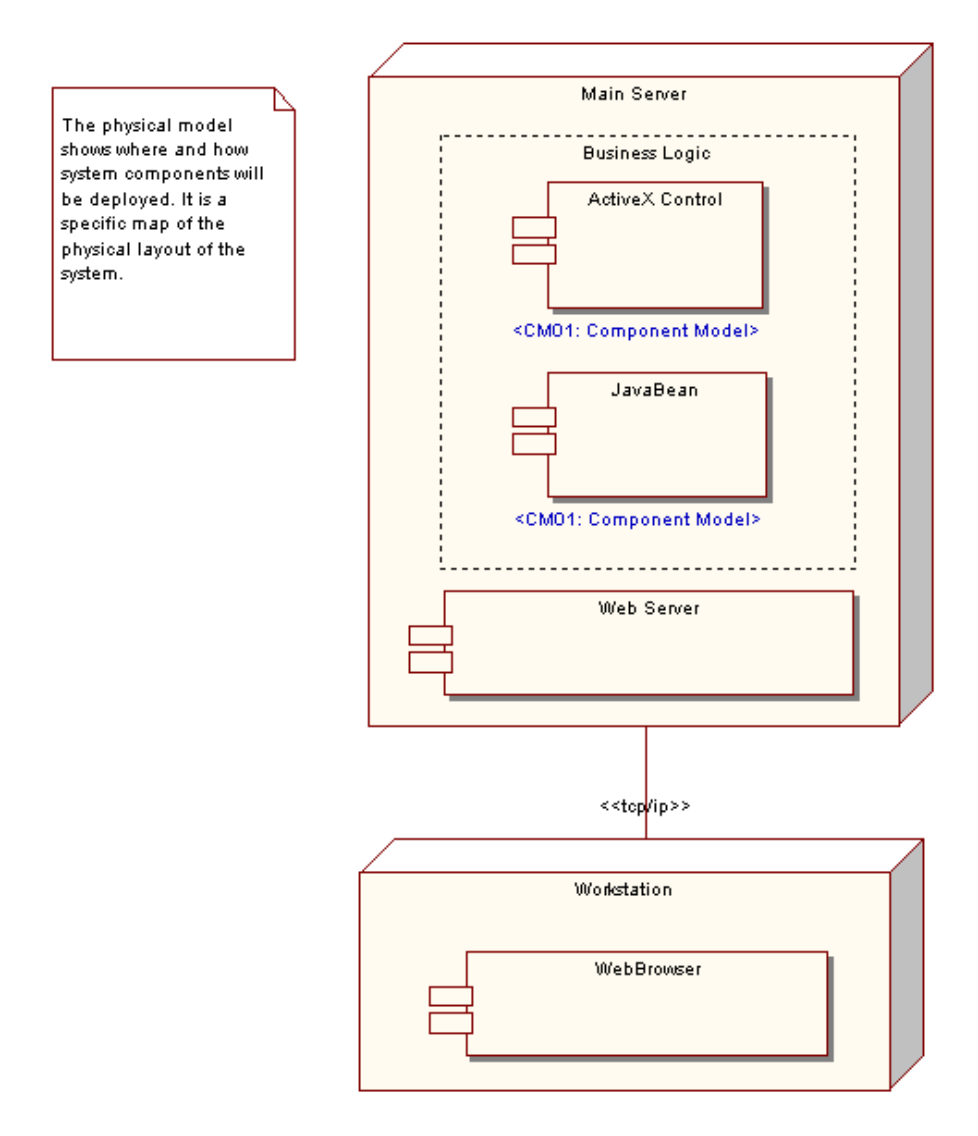
Contoh Component Diagram



Deployment Diagram

- *Deployment/physical diagram* menggambarkan detail bagaimana komponen di*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.
- Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

Contoh Deployment Diagram



Langkah-langkah Penggunaan UML

Berikut ini adalah tips pengembangan piranti lunak dengan menggunakan UML:

1. Buatlah daftar *business process* dari level tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.
2. Petakan *use case* untuk tiap *business process* untuk mendefinisikan dengan tepat fungsionalitas yang harus disediakan oleh sistem. Kemudian perhalus use case diagram dan lengkapi dengan *requirement*, *constraints* dan catatan-catatan lain.
3. Buatlah *deployment diagram* secara kasar untuk mendefinisikan arsitektur fisik sistem.

Langkah-langkah Penggunaan UML (2)

4. Definisikan *requirement* lain (non-fungsional, *security* dsb) yang juga harus disediakan oleh sistem.
5. Berdasarkan *use case diagram*, mulailah membuat *activity diagram*.
6. Definisikan objek-objek level atas (*package* atau *domain*) dan buatlah *sequence* dan/atau *collaboration diagram* untuk tiap alir pekerjaan. Jika sebuah *use case* memiliki kemungkinan alir normal dan error, buatlah satu diagram untuk masing-masing alir.
7. Buarlah rancangan *user interface* model yang menyediakan antarmuka bagi pengguna untuk menjalankan skenario *use case*.

Langkah-langkah Penggunaan UML (3)

8. Berdasarkan model-model yang sudah ada, buatlah *class diagram*. Setiap *package* atau *domain* dipecah menjadi hirarki *class* lengkap dengan atribut dan metodanya. Akan lebih baik jika untuk setiap *class* dibuat *unit test* untuk menguji fungsionalitas *class* dan interaksi dengan *class* lain.
9. Setelah *class diagram* dibuat, kita dapat melihat kemungkinan pengelompokan *class* menjadi komponen-komponen. Karena itu buatlah component diagram pada tahap ini. Juga, definisikan tes integrasi untuk setiap komponen meyakinkan ia berinteraksi dengan baik.
10. Perhalus *deployment diagram* yang sudah dibuat. Detilkan kemampuan dan *requirement* piranti lunak, sistem operasi, jaringan, dan sebagainya. Petakan komponen ke dalam node.

Langkah-langkah Penggunaan UML (4)

11. Mulailah membangun sistem. Ada dua pendekatan yang dapat digunakan :
    - Pendekatan *use case*, dengan meng-*assign* setiap *use case* kepada tim pengembang tertentu untuk mengembangkan *unit code* yang lengkap dengan tes.
    - Pendekatan komponen, yaitu meng-*assign* setiap komponen kepada tim pengembang tertentu.
12. Lakukan uji modul dan uji integrasi serta perbaiki model berserta *code*nya. Model harus selalu sesuai dengan *code* yang aktual.
13. Piranti lunak siap dirilis.

Tool yang Mendukung UML

Tool pendesainan yang mendukung UML, baik komersial maupun tidak, yaitu:

- Rational Rose (www.rational.com)
- Together (www.togethersoft.com)
- Object Domain (www.objectdomain.com)
- Jvision (www.object-insight.com)
- Objecteering (www.objecteering.com)
- MagicDraw (www.nomagis.com/magicdrawuml)
- Visual Object Modeller (www.visualobject.com)

Terima Kasih