BAB II

LANDASAN TEORI

2.1. Teori-teori Umum

2.1.1. Pengertian Sistem

Menurut Kendall (2008, p761), sistem adalah kumpulan subsistem yang saling berhubungan dan saling bergantung, yang bekerja sama untuk mencapai tujuan yang telah ditentukan sebelumnya. Semua sistem memiliki input, proses, output dan *feedback*.

Whitten (2007, p6) berpendapat bahwa sistem adalah sekumpulan komponen yang saling berhubungan yang bekerja sama untuk mencapai hasil yang diinginkan.

Menurut Stevenson (2009, p20), sistem adalah sebuah kumpulan bagian-bagian yang saling berhubungan yang harus bekerja secara bersama.

Berdasarkan definisi tersebut di atas dapat disimpulkan bahwa sistem adalah sekumpulan bagian yang saling berhubungan yang bekerja sama untuk mencapai suatu tujuan tertentu.

2.1.2. Pengertian Informasi

Menurut Bennett (2006, p15), "*information is conveyed by messages and has meaning. Meaning always depends on the perspective of the person who receive messages."*

Berdasarkan pendapat McLeod (2001, p15) informasi adalah data yang telah diproses, atau data yang memiliki arti.

Informasi adalah data yang telah diproses atau diolah ulang menjadi bentuk yang berarti. Informasi dibentuk dari kombinasi data yang diharapkan memberikan arti bagi penerimanya.

2.1.3. Pengertian Sistem Informasi

Menurut Whitten (2007, p6), sistem informasi adalah suatu pengaturan orang, data, proses, dan teknologi informasi yang berinteraksi untuk mengumpulkan, mengolah, menyimpan dan menyediakan output informasi yang dibutuhkan untuk mendukung sebuah organisasi. Dalam arti yang sangat luas, sistem informasi merupakan istilah yang sering digunakan untuk merujuk kepada interaksi antara orang-orang, proses algoritma, data dan teknologi.

Menurut Hall (2001, p7) sistem informasi adalah sebuah rangkaian prosedur formal di mana data dikumpulkan, diproses menjadi informasi dan didistribusikan kepada para pemakai.

Berdasarkan pengertian di atas dapat disimpulkan bahwa sistem informasi adalah suatu kombinasi pengaturan orang, data, proses, dan teknologi informasi yang dikembalikan kepada prosedur, mengambil, mengubah dan menyebarkan output informasi yang dibutuhkan untuk mendukung sebuah organisasi.

2.1.4. Tujuan Sistem Informasi

Tujuan sistem informasi berbeda-beda antara satu perusahaan dan perusahaan lainnya. Setiap perusahaan harus menyesuaikan sistem informasinya sesuai dengan kebutuhan perusahaan. Namun menurut Hall (2001, p18) terdapat 3 (tiga) tujuan utama yang umum bagi semua sistem, yaitu

1. Untuk mendukung fungsi kepengurusan manajemen

   Kepengurusan merujuk ke tanggung jawab manajemen untuk mengatur sumber daya perusahaan dengan benar. Sistem informasi menyediakan informasi tentang kegunaan sumber daya ke pemakai eksternal melalui laporan keuangan tradisional dan laporan-laporan yang diminta lainnya. Secara internal, pihak manajemen menerima informasi kepengurusan dari berbagai laporan pertanggung jawaban.
2. Untuk mendukung pengambilan keputusan manajemen

   Sistem informasi memberikan para manajer informasi yang mereka perlukan untuk melakukan tanggung jawab pengambilan keputusan.

3. Untuk mendukung kegiatan operasi perusahaan hari demi hari

   Sistem informasi menyediakan informasi bagi personel operasi untuk membantu mereka melakukan tugas mereka setiap hari.

2.1.5. Komponen Sistem Informasi

Menurut Turban (2003, p16) komponen-komponen dasar sistem informasi adalah sebagai berikut:

1. Perangkat keras (*Hardware*)

   Sekumpulan peralatan seperti prosesor, monitor, keyboard, dan printer yang menerima data dan informasi, memroses serta menampilkannya.

2. Perangkat lunak (S*oftware*)

   Sekumpulan program komputer yang memungkinkan perangkat keras (*hardware*) untuk memroses data.

3. Basis data (*Database*)

   Kumpulan dari *file, record*, dan lain-lain yang terorganisir dimana berguna untuk menyimpan data dan hubungannya.

4. Jaringan (*Network*)

   Suatu sistem yang terhubung dimana menyediakan penggunaan secara bersama-sama sumber daya antar komputer yang berbeda-beda.

5. Prosedur (*Procedures*)

   Strategi kebijakan, metode, dan aturan untuk menggunakan sistem informasi.

6. Personil (*People*)

   Merupakan elemen yang paling penting dalam sistem informasi, yang terdiri dari mereka yang bekerja dengan sistem informasi itu sendiri atau menggunakan keluarannya (output).

2.1.6. Pengertian Data

Menurut Hoffer (2009, p46), data adalah gambaran objek dan peristiwa yang memiliki arti dan kegunaan dalam lingkungan pengguna.

Berdasarkan Whitten (2007, p21) data adalah fakta mentah (belum diolah) mengenai orang, tempat, kejadian dan hal yang berkepentingan dengan organisasi.

Dari definisi tersebut dapat disimpulkan bahwa data adalah fakta mengenai suatu hal yang berkepentingan bagi organisasi.

2.1.7. Pengertian Basis Data

Menurut McLeod (2001, p258), basis data adalah suatu koleksi data komputer yang terintegrasi, diorganisasikan, dan disimpan dalam suatu cara yang memudahkan pengambilan kembali. Basis data digunakan untuk membantu perusahaan, atau pemakai untuk mengolah informasi yang dibutuhkan secara cepat dan mudah, baik untuk memasukan dan menyimpan informasi maupun memperbaiki informasi yang sudah ada dan menampilkannya ke dalam berbagai

bentuk atau format yang tepat dan sangat membantu, misalnya dalam bentuk laporan, grafik maupun dalam bentuk peta, yang dapat menghasilkan suatu ringkasan yang memberikan kemudahan bagi pemakai mengenai sistem terkini.

Menurut Whitten (2007, p518), *database* adalah kumpulan *file* yang saling terkait. Basis data tidak hanya merupakan kumpulan *file*. Catatan dalam *file* masing-masing harus memungkinkan berhubungan (menganggap *database* sebagai "*pointer*") untuk catatan dalam *file* lain. "*a database is a collection of interrelated files. A database is not merely a collection of file. The record in each file must allow for relationships (think of them as "pointers") to the records in other files"*.

Basis data adalah kumpulan *file* yang saling terkait yang disimpan dan diatur dalam suatu cara tertentu yang dapat memudahkan pengguna untuk mengaksesnya kembali.

2.2. Teori - teori Khusus

2.2.1. Sistem Persediaan

Menurut Epstein (2004, p208*)* persediaan dapat didefinisikan sebagai berikut: "Persediaan merupakan aset pengadaan barang di dalam sebuah bisnis, atau yang sedang dalam proses produksi untuk penjualan tertentu, atau dalam wujud material atau pendukung untuk digunakan dalam proses produksi atau penyumbangan jasa."

Menurut Stevenson (2009, p571) *re-order point* terjadi ketika jumlah persediaan telah mendekati jumlah minimal yang telah ditetapkan. Jumlah tersebut biasanya mencakup permintaan pada waktu nyata. S*erver* disini bertugas untuk menanggulangi kemungkinan kehabisan stok selama waktu tertentu. Untuk mengetahui apakah persediaan telah menyentuh stok minimal, maka perhitungan persediaan sangat diperlukan.

Pada dasarnya, arti dari persediaan adalah simpanan material yang berupa bahan mentah, barang dalam proses dan barang jadi. Setiap perusahaan, baik itu perusahaan perdagangan ataupun perusahaan jasa selalu mengadakan persediaan. Tanpa adanya persediaan, para pengusaha akan dihadapkan pada resiko bahwa perusahaannya pada suatu waktu tidak dapat memenuhi keinginan pelanggan yang memerlukan atau meminta barang atau jasa.

Dengan demikian dapat disimpulkan bahwa persediaan adalah barang-barang yang dibeli untuk produksi, serta dijual kembali dan habis dipakai oleh perusahaan. Karena pentingnya peranan persediaan barang bagi perusahaan, kekayaan tersebut haruslah dikelola dengan baik dan selalu berada dalam pengendalian dan pengawasan yang ketat.

Persediaan dalam sebuah perusahaan harus dikelola dengan baik karena:

1. Persediaan merupakan investasi yang membutuhkan modal besar.
2. Mempengaruhi pelayanan ke pelanggan.

3. Mempunyai pengaruh pada fungsi operasi, pemasaran, dan fungsi keuangan.

Sedangkan tujuan pengawasan persediaan adalah sebagai berikut:

1. Menjaga agar jangan sampai kehabisan persediaan.
2. Menstabilkan pembentukan persediaan.

2.2.2. Sistem Informasi Persediaan

Chase (1998, p546) menyatakan bahwa sistem informasi persediaan adalah kesatuan kebijakan dan pengendalian yang mengawasi tingkat persediaan dan menentukan kapan tingkat persediaan diperiksa, kapan persediaan harus dipenuhi kembali dan berapa besar jumlah pembelian yang harus dilakukan.

Sedangkan Mulyadi (2002, p555) berpendapat bahwa sistem informasi persediaan adalah suatu sistem yang menyediakan informasi atau laporan-laporan yang dibutuhkan oleh pihak manajemen yang berhubungan dengan operasi pemesanan, penyimpanan dan persediaan bahan baku.

Penulis menyimpulkan bahwa sistem informasi persediaan adalah suatu pengaturan orang, data, proses, dan teknologi informasi yang berinteraksi untuk mengumpulkan, mengolah, menyimpan dan menyediakan output informasi yang dibutuhkan untuk mendukung perusahaan dalam menyimpan material yang berupa bahan mentah, barang dalam proses dan barang jadi.

2.3. Analisis dan Perancangan Sistem Informasi

Analisis dan perancangan sistem informasi merupakan tahap penting di dalam pengembangan sistem informasi, dimana semua permasalahan pada sistem berjalan diidentifikasi untuk dapat menentukan perbaikan dan pemecahan masalah tersebut. Berdasarkan keputusan yang diambil, analis akan merancang sistem baru yang diharapkan dapat mengatasi permasalahan pada sistem sebelumnya dan dapat memenuhi kebutuhan pengguna.

2.3.1. Analisis Sistem

Analisis sistem menurut Turban (2009, p304) adalah penelitian atas permasalahan organisasi yang direncanakan untuk diatasi dengan sistem informasi. Pada tahap ini permasalahan bisnis lebih digambarkan secara detail, penyebabnya diidentifikasi, mencari solusi yang dapat dilakukan dan mengidentifikasi apa saja yang harus dicapai dari solusi tersebut.

Sedangkan McLeod (2001, p190) mendefinisikan analisis sistem sebagai penelitian atas sistem yang telah ada dengan tujuan untuk merancang atau untuk memperbaharui sistem tersebut.

Dengan demikian, dapat disimpulkan bahwa analisis sistem adalah penelitian atas sistem yang berjalan untuk mendapatkan kekurangan dan juga

permasalahan yang ada untuk selanjutnya dijadikan tujuan atau dasar spesifikasi untuk sistem yang akan dikembangkan.

2.3.2. Perancangan Sistem

Menurut Mulyadi (2002, p51), perancangan sistem adalah proses penerjemahan kebutuhan pemakai informasi ke dalam alternatif rancangan sistem informasi yang digunakan kepada pemakai informasi untuk dipertimbangkan.

Sedangkan Turban (2009, p304) berpendapat perancangan sistem menggambarkan bagaimana sistem dapat memenuhi tujuan atau target. Perancangan sistem memiliki 2 (dua) aspek yaitu

- *Logical systems design*: apa yang akan sistem lakukan, menggunakan spesifikasi abstrak
- *Physical system design*: bagaimana sistem akan menjalankan fungsinya, dengan menggambarkan spesifikasi secara benar.

Dapat ditarik kesimpulan bahwa perancangan sistem adalah proses penerjemahan kebutuhan pemakai informasi ke dalam rancangan sistem informasi untuk memberikan gambaran yang jelas akan sistem informasi tersebut.

2.3.2.1. *Prototyping*

Menurut Turban (2009, p416) *prototyping* adalah suatu pengembangan dan pengujian sebuah model atau *prototype* dari suatu aplikasi dalam beragam tampilan dan fungsi yang interaktif dan iteratif dari suatu sistem bisnis.

2.3.2.2. *User Interface Design*

Turban (2009, p418) berpendapat bahwa kegiatan desain *user interface* fokus kepada mendukung interaksi antara pengguna dan aplikasi sistemnya. Perancang berkonsentrasi pada desain form input dan output yang interaktif dan menarik serta mudah digunakan.

2.4. Analisis dan Perancangan Berorientasi Obyek

Object-Oriented Analysis and Design (OOAD) adalah metode untuk menganalisa dan merancang sistem dengan pendekatan berorientasi obyek menurut Mathiassen (2000, p135). Obyek diartikan sebagai suatu entitas yang memiliki identitas, *state*, dan *behavior* menurut Mathiassen (2000, p4). Pada analisa, identitas sebuah obyek menjelaskan bagaimana seorang *user* membedakannya dari obyek lain, dan *behavior object* digambarkan melalui *event* yang dilakukannya. Sedangkan pada perancangan, identitas sebuah obyek digambarkan dengan cara bagaimana obyek lain mengenalinya sehingga dapat

diakses, dan *behavior object* digambarkan dengan *operation* yang dapat dilakukan obyek tersebut yang dapat mempengaruhi obyek lain dalam sistem.

2.4.1. Keuntungan OOAD

Keuntungan dari OOAD menurut Mathiassen (2000,p5) adalah:

1. Merupakan konsep umum yang dapat digunakan untuk memodelkan hampir semua fenomena dan dapat dinyatakan dalam bahasa umum (*natural language*)

    - *Noun* menjadi *object* atau *class*
    - *verb* menjadi *behavior*
    - *Adjective* menjadi *attributes*

2. Memberikan informasi yang jelas tentang *context* dari sistem
3. Mengurangi biaya *maintenance*
4. Memudahkan untuk mencari hal yang akan diubah
5. Membuat perubahan menjadi lokal tidak berpengaruh pada modul yang lain.

2.4.2. Prinsip Umum OOAD

Menurut Mathiassen (2000, p18) terdapat empat prinsip umum OOAD, yaitu:

1. *Model the context*

   Sistem yang bermanfaat sesuai dengan konteks OOAD. Maka adalah penting untuk model kedua-duanya baik *application domain* dan *problem domain* selama analisis dan desain.

2. *Emphasize the architecture*

   Merupakan arsitektur yang mudah dipahami yang memfasilitasi kolaborasi antara *designer* dan *programmer*. Arsitektur yang fleksibel membuat modifikasi dan perbaikan sistem yang lebih baik.

3. *Reuse Patterns*

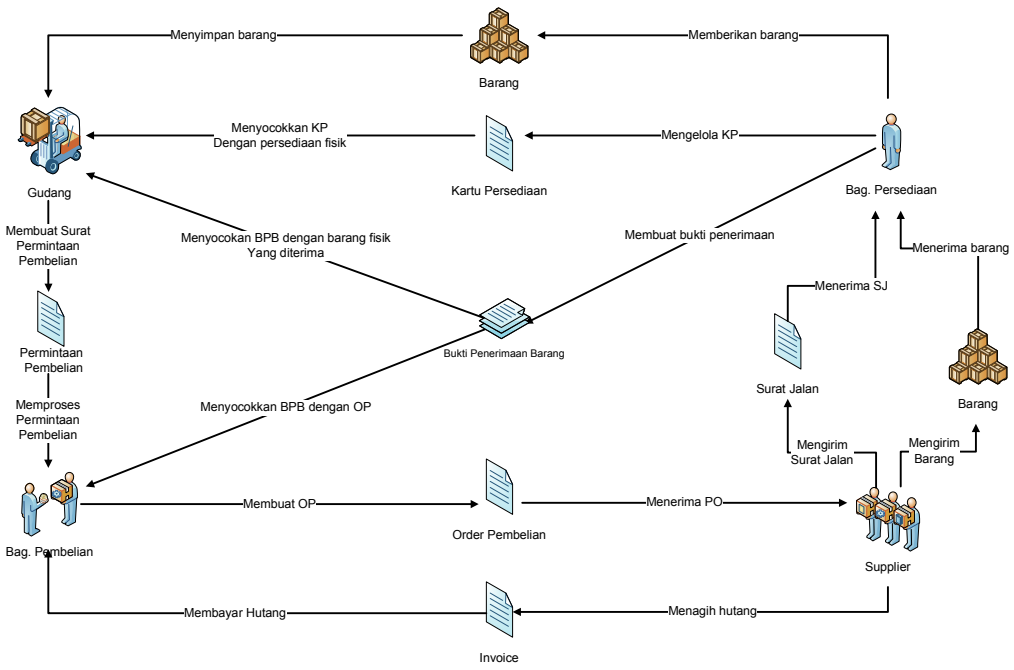
   Dibangun berdasarkan gagasan-gagasan yang kuat dan komponen *pretested* memperbaiki kualitas sistem dan produktivitas dari proses *development.*

4. *Tailor the method to suit specific projects*

   Setiap usaha *development* masing-masing mempunyai tantangan yang unik. OOAD harus disesuaikan dengan kebutuhan-kebutuhan yang khusus dari situasi analisis dan desain yang diberikan.

2.4.3. *System Definition*

Menurut Mathiassen (2000, p23), *system definition* merupakan sebuah penjelasan singkat dari sistem terkomputerisasi yang diungkapkan dengan bahasa alamiah. Di dalam *system definition* juga dijelaskan tentang *rich picture* dan *FACTOR*.



Gambar 2.1 *Rich picture*

Rich picture merupakan sebuah penggambaran secara tidak formal yang menggambarkan pemahaman *illustrator* dari suatu situasi. *Rich picture*

memfokuskan pada aspek penting dari suatu situasi yang mana ditentukan oleh *illustrator*.

FACTOR terdiri dari enam bagian yaitu:

- *Functionality* : fungsi dari sistem yang mendukung *application domain.*
- *Application Domain* : bagian dari organisasi yang mengadministrasi, memonitoring, dan mengontrol *problem domain.*
- *Condition* : kondisi dari sistem yang dikembangkan dan digunakan.
- *Technology* : teknologi yang digunakan untuk mengembangkan sistem dan teknologi apa yang akan dijalankan.
- *Objects* : objek dasar dari *problem domain*.
- *Responsibility* : tanggung jawab sistem secara keseluruhan dalam hubungannya dengan konteks.

2.5. *Problem Domain*

2.5.1. *Object, Attribute, Method, Encapsulation*

Menurut Mathiassen (2000, p4), objek adalah suatu entitas yang mempunyai identitas, *state* dan *behavior*. Contohnya: pelanggan, karyawan. Objek *instance* adalah setiap hal yang khusus misalnya orang, tempat benda, sebagai nilai atribut dari objek.

Atribut adalah data yang dipresentasikan serta memiliki karakteristik yang menarik dari sebuah objek. Menurut Mathiassen (2000, p92), atribut adalah properti deskriptif dari sebuah kelas atau *event*. Contoh seperti pelanggan yang memiliki nama, alamat, nomor identitas pelanggan disebut sebagai atribut.

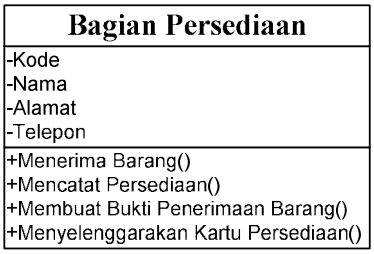
Behavior adalah apa yang dapat dilakukan suatu objek. Dalam analisis berorientasi objek, *behavior* dari objek biasanya mengacu pada *method, operation* dan *service. Encapsulation* adalah penggabungan dari beberapa *item* bersama-sama ke dalam satu unit.

Menurut (Mathiassen, 2000, p46) *problem domain* adalah bagian dari sebuah konteks yang diadministrasikan, dimonitor atau dikontrol oleh sebuah sistem.

Problem domain terdiri dari :

1. *Classes*

   Class adalah sebuah deskripsi dari kumpulan objek-objek yang memiliki struktur *behavior pattern* dan atribut yang sama.



Gambar 2.2 *Class*

Menurut Mathiassen (2000, p49), abstraksi, klasifikasi, dan seleksi adalah tugas utama dalam aktifitas kelas. Kelas merupakan tujuan utama dalam mendefinisikan dan membatasi problem domain. Kelas terdiri dari tiga bagian yaitu:

a. Nama kelas yaitu yang mendefinisikan kelas itu sendiri.
b. Atribut

   Atribut memiliki beberapa sifat antara lain:

   - *Private*

     Private memiliki sifat yang tidak bisa dipanggil dari luar kelas itu sendiri.

   - *Protected*

     Merupakan suatu sifat yang hanya dapat dipanggil dikelas itu sendiri dan dan hanya bisa diwarisi pada sub kelas yang bersangkutan.

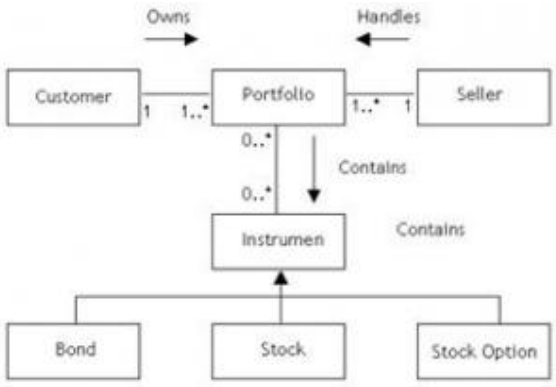
- *Public*

  Public merupakan sebuah sifat dalam kelas yang dapat dipanggil oleh dan digunakan oleh kelas yang lain

c. Operasi

   Merupakan sesuatu kegiatan yang dilakukan oleh sebuah kelas.

Menurut Mathiassen (2000, p69), *class diagram* memberikan gambaran *problem domain* dengan menggambarkan hubungan secara struktural antara objek kelas dan objek di dalam model tersebut. Contoh *class diagram* dapat dilihat pada gambar 2.3.



Gambar 2.3 *class diagram*

2. *Structure*

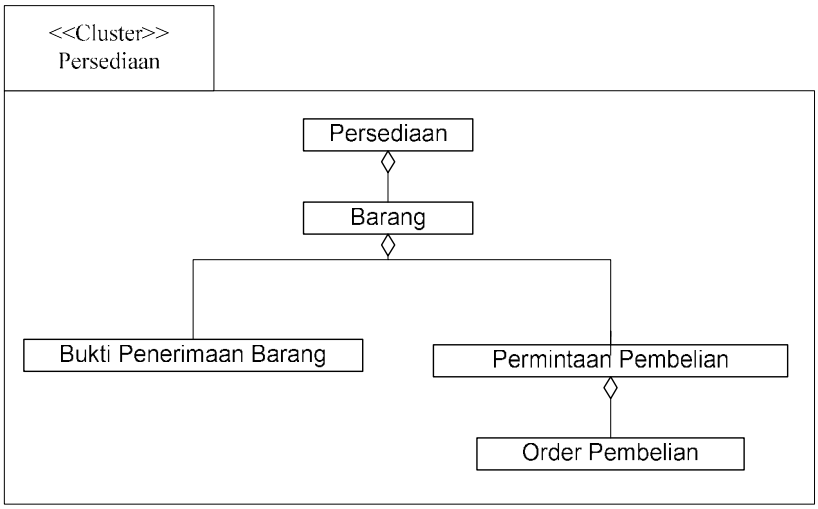
Aktifitas *structure* difokuskan pada hubungan antara *classes* dan objek. Menurut Mathiassen (2000, p72) struktur antar *class* terdiri dari dua tipe yaitu :

a. Struktur generalisasi

   Generalisasi adalah *super class* yang menjelaskan *sub class*

b. Struktur *Cluster*

   Cluster adalah sebuah kumpulan dari *classes* yang berhubungan. Kelas didalam *Cluster* biasanya berhubungan secara struktur generalisasi atau struktur agregasi.



Gambar 2.4 *Cluster*

Sedangkan struktur antar obyek menurut Mathiassen (2000, p75) terdiri dari dua tipe yaitu:

a. Struktur agregasi

   Struktur agregasi adalah sebuah hubungan antara dua atau lebih objek.

b. Struktur asosiasi

   Struktur asosiasi adalah sebuah hubungan antara dua atau lebih objek tetapi berbeda dengan agregasi di mana hubungan objek-objek yang terasosiasi tersebut tidak mendefinisikan *property* dari suatu objek.

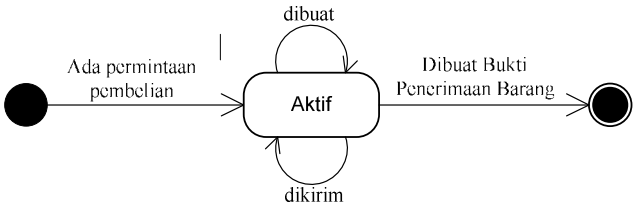
3. *Events*

Berdasarkan pendapat Mathiassen (2000, p51), e*vent* adalah sebuah kejadian seketika yang melibatkan satu atau lebih objek. *Event table* mempermudah dalam menganalisa sistem agar tidak ada *event* yang terlupakan dalam membuat suatu *class diagram*.

4. *Behaviour Pattern*

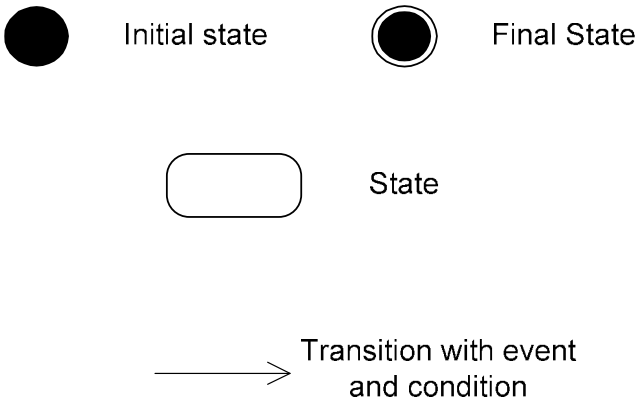
Menurut Mathiassen (2000, p90) *behaviour pattern* adalah deskripsi dari *event trace* yang mungkin untuk semua objek di dalam *class*. *Event trace* adalah urutan *event-event* dari suatu objek tertentu. *Behavior pattern* dapat digambarkan dalam *state diagram*.

State Diagram menggambarkan behavior umum dari semua objek dari *class* tertentu, yang terdiri dari bagian-bagiannya dan transisi di antaranya dan juga dapat menjelaskan *usecase*. *Statechart diagram* menggambarkan transisi dan perubahan keadaan suatu objek pada sistem sebagai akibat dari stimulasi yang diterima.



Gambar 2.5 *Statechart*

Notasi pada *statechart diagram* adalah sebagai berikut:



Gambar 2.6 Notasi *Statechart Diagram*

Notasi pada *behavioral pattern* terdiri dari tiga macam yaitu:

a. *Sequence* merupakan *events* yang terjadi sekali saja
b. *Selection* merupakan sesuatu yang keluar dari peristiwa yang terjadi
c. *Iteration* merupakan *events* yang terjadi nol atau lebih

2.5.2. Application Domain

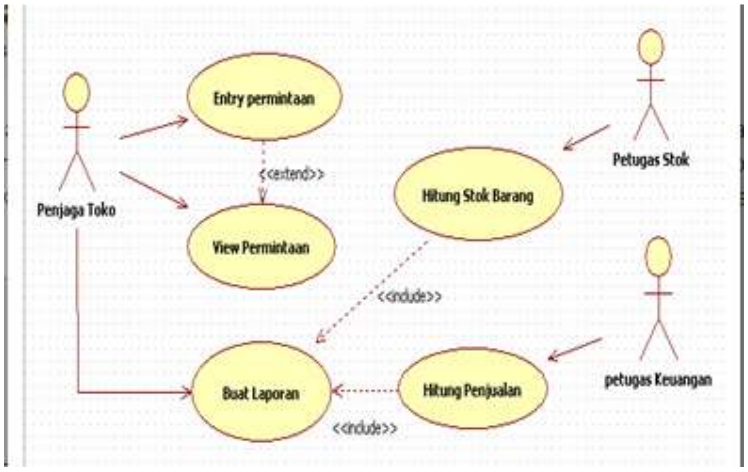
Application Domain adalah organisasi yang mengelola, memantau atau mengontrol *problem domain***.** *Application Domain* terdiri dari :

1. *Usage*

Usage menggambarkan bagaimana sistem berinteraksi dengan orang dan sistem di dalam konteks. *Usage* terdiri dari *sequence* diagram dan *use case diagram*.

a. *Use case*

Use case diagram seperti dalam gambar 2.6 menurut Satzinger (2005, p213) adalah sebuah diagram yang menunjukkan bermacam-macam peran pengguna dan cara mereka berinteraksi dengan sistem.



Gambar 2.7 *Use case diagram*

Actor menurut Mathiassen (2000, p119) adalah abstraksi dari *user* atau sistem lain yang berinteraksi dengan sistem target. *Actor* merupakan abtraksi orang-orang dan sistem yang lain yang aktif pada sistem *function*. Sebuah *use case* adalah abtraksi dari interaksi dengan target sistem. *Use case* dapat diaktifkan oleh *actor* atau oleh target sistem. *Use case* yang lengkap menentukan semua penggunaan target sistem di dalam *application domain.*

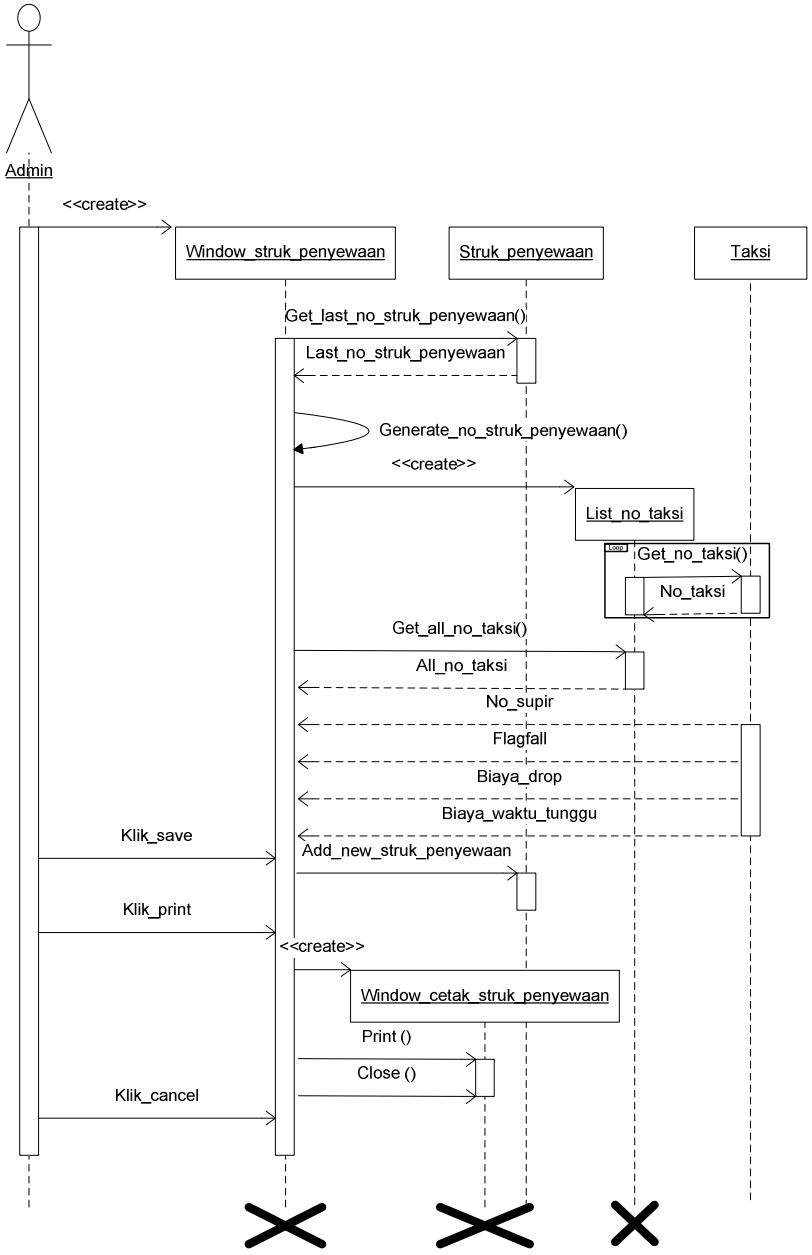
b. *Sequence*

Menurut Bennett (2006, p252) *Sequence diagram* merupakan peralatan untuk interaksi berkomunikasi diagram. Sebuah interaksi didesain antara objek atau sistem yang berpartisipasi dalam sebuah

kolaborasi. Interaksi dijelaskan oleh pesan-pesan yang diletakkan pada sebuah waktu atau lebih dari dua pesan yang akan dikirim pada saat yang sama. Interaksi merupakan peran komunikasi yang penting. Beberapa konsep dapat diterapkan pada konteks yang bervariasi.

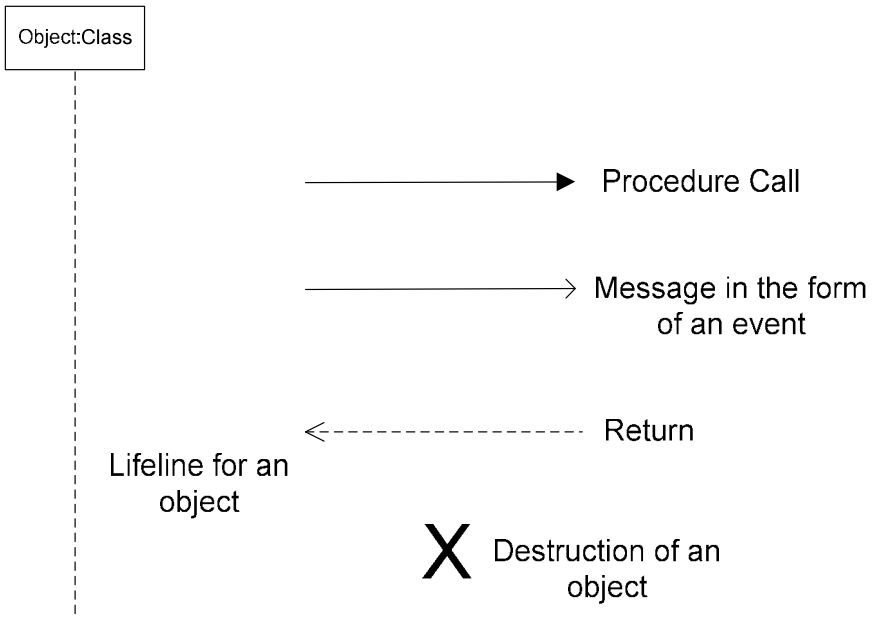
Menurut Mathiassen (2000, p340) s*equence diagram* menggambarkan interaksi antara objek secara beraturan sesuai dengan waktu. *Sequence diagram* dapat digambarkan dalam beberapa level secara detail dan untuk tujuan yang berbeda pada beberapa langkah yang dikembangkan secara *lifecycle.* Ketika pesan dikirim pada sebuah objek akan meminta sebuah operasi dari objek. Nama pesan biasanya sesuai dengan operasi yang akan diminta. Sebuah pesan diterima, operasi yang telah diminta akan melaksanakan pesan tersebut. Pada beberapa tahap selama operasi yang dilakukan tersebut disebut sebagai aktivasi.

Periode aktivasi pada periode termasuk beberapa rentang waktu selama operasi yang menunggu respon dari operasi yang lain bahwa yang diminta akan segera dilaksanakan. Berikut ini merupakan tampilan *sequence diagram* menurut *Simon Bennett* :



Gambar 2.8 *Sequence Diagram*

Notasi yang terdapat dalam *sequence diagram* dapat dilihat pada gambar 2.9.



Gambar 2.9 Notasi *Sequence Diagram*

2. *Function*

Function menurut Mathiassen (2000, p138) adalah sebuah fasilitas untuk membuat sebuah model yang berguna bagi *actor*. *Function* terdiri dari *complete function list*. Ada beberapa tipe dari *function* antara lain:

a. *Update Function* diaktifkan oleh sebuah event dari *problem domain* dan hasilnya adalah sebuah perubahan pada *model state.*
b. *Signal Function* diaktifkan oleh sebuah perubahan pada *model state* dan hasilnya sebuah reaksi pada konteks. Reaksi ini mungkin sebuah tampilan kepada *actor* di *application domain.*
c. *Read Function* diaktifkan oleh sebuah kebutuhan informasi pada sebuah tugas kerja pada *actor* dan hasilnya tampilan sistem yang berhubungan dengan bagian *model*.
d. *Compute Function* diaktifkan oleh sebuah kebutuhan informasi pada sebuah tugas kerja pada *actor* dan terdiri dari sebuah perhitungan yang melibatkan informasi yang disediakan oleh *actor* atau *model.* Hasilnya sebuah tampilan hasil perhitungan.

3. *Interface*

Interface menurut Mathiassen (2000, p151) adalah fasilitas-fasilitas yang membuat model dan fungsi-fungsi tersedia bagi *actor*. Hasil dari *interfaces* adalah *user interface* dan *system interface*.

User interface adalah *style dialog* dan bentuk-bentuk presentasi, daftar elemen dari *user interface* yang lengkap, *windows diagram* yang dipilih dan *navigation diagram.* Contoh *user interface* dapat dilihat pada gambar 2.10.

Gambar 2.10 *User Interface*

System interface adalah *class diagram* untuk eksternal *device* dan protokol-protokol untuk interaksi dengan sistem lain. *Navigation diagram* merupakan semua window dari *user interface* dan hubungan dinamiknya.

2.5.3. *Architectural Design*

Menurut Mathiassen (2000, p177), aktivitas dalam *architectural design*:

a. *Criteria*

Criteria merupakan sebuah properti dari sebuah arsitektur. Kriteria umum disain:

1. *Usable* : kesesuaian sistem dalam organisasi, hubungan kerja dan konteks teknis.
2. *Flexible* : biaya yang dibutuhkan untuk memodifikasi sistem yang dikembangkan.
3. *Comprehensible* : usaha yang dibutuhkan untuk memperoleh pemahaman yang masuk akal dari sebuah sistem.
4. *Secure* : adanya autorisasi pada saat menggunakan data dan fasilitas sistem tersebut.
5. *Efisien* : sistem yang sudah ada dan dapat secara cepat sesuaia dengan fungsinya.
6. *Correct* : kemampuan dari sistem yang ada dan memberikan kepuasan kepada pengguna
7. *Testable* : sistem yang sudah ada mudah ditelusuri dan dapat mudah diketahui secara otomat

8. *Realible* : menjaga tingkat *performance* pada waktu dan situasi tertentu.
9. *Reusable* : mempermudah perubahan dari sistem yang lama ke sistem yang baru.
10. *Maintainable* : pengalokasian biaya dan perbaikan sistem yang rusak
11. *Portable* : biaya dari pemindahan suatu sistem ke sistem *platform* yang lainnya.
12. *Interoparable* : biaya dari *coupling* sistem dengan sistem lainnya.

b. Komponen arsitektur

Komponen arsitektur merupakan struktur sistem, yang saling berhubungan dengan komponen-komponen. Komponen adalah sebuah kumpulan dari bagian-bagian program yang membentuk sebuah kesatuan dan mempunyai tugasnya masing-masing.

c. Proses

Proses menggambarkan bagaimana proses sistem didistribusi dan dikoordinasi. Arsitektur proses: struktur eksekusi sistem diubah pada *independent* proses. Tujuan dari desain proses arsitektur adalah untuk menyusun eksekusi pada *level physical*.

2.5.3.1. *Component Design*

Menurut Mathiassen (2000, p232), ada beberapa aktifitas dalam *component design* antara lain:

a. *Model Component*

Model component adalah sebuah bagian dari sebuah sistem yang mengimplementasikan problem domain model. Tujuannya adalah mengirim data yang sekarang dan yang lalu kepada fungsi, *interfaces* dan kepada *user* dan sistem yang lain. Hasil dari aktivitas *model component* adalah sebuah versi revisi pada *class diagram* dari aktivitas analisis. Dalam desain analisis model digambarkan sebagai kelas diagram yang dikombinasikan dengan *statechart* diagram untuk setiap kelas. Dalam *model component* desain lebih fokus pada informasi yang diambil dari *event.*

Untuk menentukan *Revised Class*, pertama-tama harus ditentukan *private* dan *common event*. *Private event* adalah *event* yang hanya melibatkan hanya satu abjek pada *problem domain*. *Common event* adalah adalah *event* yang melibatkan beberapa objek, event ini menggambarkan hubungan antara satu objek dengan objek lainnya dan

memungkinkan menambahkan hubungan struktural, untuk memungkinkan objek lain mengakses atributnya.

Untuk *event* yang terjadi sekali tampilkan *event* tersebut sebagai atribut pada kelas dimana *event* itu berada. *Event* ini ditandai dengan ” + ”. Untuk *event* yang terjadi berkali-kali atau iterasi keluarkan *event* ini sebagai kelas baru. *Event* ini ditandai dengan ” * ”. Hubungkan kelas baru dengan *class* dari mana *event* itu berasal dengan hubungan agregasi.

b. *Function Component*

Menurut Mathiassen (2000, p252) merupakan sebuah bagian dari sistem yang mengimplementasikan kebutuhan-kebutuhan fungsional. Tujuan dari *function component* adalah untuk memberikan *user interfaces* dari *system components* yang diakses kedalam model.

Hasil dari *function component* adalah sebuah kelas diagram dengan operasi dan spesifikasi operasi yang kompleks.

Tabel 2.1 *Operation Specification*

Name	***Register transactions***
Category	*Active update* *Passive read,compute,signal*
Purpose	Membuat transaksi baru
Input data	Nomor *account,* tanggal, jumlah
Conditions	Objek dalam kelas *account* dengan memberikan nomor *account*
Effect	Sebuah objek kelas transaksi
Algorithm	*Algorithm, program*
Data structure	Tipe data yang digunakan
Placement	*Accounts*
Involved object	*Account, transactions*
Triggering events	Jumlah deposito, jumlah penarikan

c. *Connecting Component*

Connecting component terbagi dua yaitu kopling (*coupling)* dan kohesi (*cohesion).* Menurut Mathiassen (2000, p272) *coupling* adalah sebuah ukuran seberapa dekat dua kelas atau komponen dihubungkan sedangkan c*ohesion* adalah sebuah ukuran seberapa baik sebuah kelas atau komponen digabungkan bersama.

2.6. *Unified Modeling Language*

Unified Modeling Language (UML) adalah suatu bahasa visual serba guna yang digunakan untuk menjelaskan, menvisualisasikan, membangun, dan mendokumentasikan suatu sistem. *Unified Modeling Language* digunakan untuk memahami, merancang, mengonfigurasi, memelihara dan mengontrol informasi tentang suatu sistem (Booch, 2007, p13).

UML terdiri dari beberapa elemen yang membentuk diagram dengan aturan tertentu. Diagram ini bertujuan untuk menggambarkan sistem dari berbagai sudut pandang.

Beberapa diagram *Unified Modeling Language* antara lain:

1. *Class Diagram*

   Class adalah suatu kategori atau kumpulan dari benda atau objek yang memiliki atribut dan operasi yang dikerjakan oleh benda atau objek tertentu. *Class diagram* merupakan diagram struktural yang menggambarkan sekumpulan *class, interface, collaboration* dan relasinya.

2. *Use Case Diagram*

   Use Case diagram merupakan gambaran dari suatu urutan yang dilakukan oleh sistem yang dilihat dari sudut pandang pelaku atau *actor*. *Use Case diagram* adalah diagram struktural yang menggambarkan sejumlah *use case* dan pelaku serta relasinya.

3. *Activity Diagram*

   Activity diagram menggambarkan aktivitas atau proses yang terjadi dalam suatu *use case* atau suatu operasi yang terjadi secara berurutan. *Activity Diagram* adalah diagram perilaku yang menggambarkan *state machine* dengan penekanan pada urutan waktu dari pesan.

4. *Sequence Diagram*

   Sequence diagram menggambarkan sebuah objek dan *message* yang dikirim dan diterima oleh objek tersebut berdasarkan urutan waktu. *Sequence diagram* adalah diagram perilaku yang menggambarkan interaksi dengan penekanan pada urutan waktu dari pesan.

5. *Statechart Diagram*

   Statcechart diagram menjelaskan behavior umum dari semua objek yang ada di dalam suatu *class* dan terdiri dari *state* dan transisi. Untuk menggambarkan objek transisinya berupa event. Statechart diagram adalah diagram perilaku yang menggambarkan state machine dengan penekanan pada urutan *event* perilaku objek. *Statechart* diagram dapat juga digunakan untuk menjelaskan *use case* dimana transisinya dijelaskan oleh *action*.

6. *Object Diagram*

   Object diagram menunjukkan sekumpulan objek dan hubungannya. *Object diagram* adalah diagram struktural yang menggambarkan

objek-objek dan relasi antar objek dalam sistem. Diagram ini menempatkan tampilan desain yang statis.

7. *Collaboration Diagram*

   Collaboration diagram adalah diagram yang menggambarkan interaksi dengan penekanan pada struktur dari objek yang mengirim dan menerima pesan. *Collaboration* diagram sifatnya *isomorphic*, artinya kita dapat mengambil suatu objek dan mengubahnya menjadi yang lain.

8. *Component Diagram*

   Component diagram adalah struktural yang menggambarkan komponen-komponen dan relasi-relasi antar komponen. Menunjukkan organisasi dan *dependency* antara sekumpulan komponen. Diagram ini menempatkan tampilan implementasi statis dari sistem. Mereka berkaitan pada *class* diagram dan komponen utama pada satu atau lebih *class, inteface,* atau *collaboration*.

9. *Deployment Diagram*

   Deployment diagram adalah diagram struktural yang menggambarkan nodes komponen dan relasi *nodes*. Menunjukkan konfigurasi dari *nodes* proses *run time* dan komponen yang terdapat di dalamnya. Diagram ini menempatkan tampilan deployement statis dari arsitektur.

Mereka terkait pada diagram komponen pada sebuah *node* khususnya membatasi satu atau lebih komponen.